scSetFormer: A Set Transformer-based framework for conditional scRNA-seq data generation

Maha Alissa Alkhalaf [1] **Sebastian Bischoff** [2 3] **Jakob H. Macke** [2 4] **Manfred Claassen** [5 6]

Abstract

scRNA-seq data provides information on the gene expression levels of a tissue at an individual cell level, thereby enabling the analysis of variability across cells of different types or conditions. scRNA-seq is, however, a time-consuming process that can take weeks to generate final data. Furthermore, rare or small cell populations complicate drawing statistically strong conclusions when compared to cell populations of a larger size. Augmentation of such cell populations via traditional upsampling techniques has been shown to worsen downstream analyses. More time-efficient and more sophisticated augmentation approaches, e.g. using generative machine learning models, are, therefore, sought. In this work, we introduce a Set Transformer-based variational autoencoder—scSetFormer—for conditional scRNA-seq data generation. Our model leverages the self-attention mechanism to capture the gene expression correlations across genes within and across cells. The integration of set attention modules makes scSetFormer permutation invariant which renders the generation process independent of the gene input order. Furthermore, scSetFormer is designed to generate cells conditioned on not only one but multiple input cells, thereby allowing it to aggregate information across different cells. Moreover, scSetFormer is flexible and builds a foundation for different *in silico* scRNA-seq data generation tasks. scSetFormer shows comparable performance and sometimes outperforms current scRNA-seq data generative models on various benchmarking metrics.

[1]Bioinformatics master thesis from November 6th, 2023 to April 9th, 2024 [2]Machine Learning in Science, University of Tübingen and Tübingen AI Center, Tübingen, Germany [3]Thesis supervisor [4]First examiner [5]Clinical Bioinformatics & Machine Learning in Translational single-cell Biology, University Hospital of Tübingen, Tübingen, Germany [6]Second examiner. Correspondence to: Maha Alissa Alkhalaf <maha.alissa-alkhalaf@student.uni-tuebingen.de>.
Copyright 2024 by the author(s).

1. Introduction

Over the past decade, single-cell RNA sequencing (scRNA-seq) has become a cornerstone of any scientific discipline that studies the behaviour of individual cells. scRNA-seq enables insights into the gene expression within up to several thousand of cells isolated from a tissue of interest at the same time (Chen et al., 2019; Kulkarni et al., 2019; Jovic et al., 2022). One of its main applications is to study the effect of a particular drug on either healthy or diseased, e.g. cancerous, cells (Wen et al., 2022). Moreover, scRNA-seq allows the understanding of the cellular developmental processes, such as the development of embryonic cells (Li, 2023) or the transition of T cells from naive to memory T cells during an infection (Efremova et al., 2020). Despite its valuable advantages, scRNA-seq is a very time-consuming process (Hahaut et al., 2022). Drawing strong conclusions about rare cell types is additionally challenging due to their small number. Isolating more cells from the respective tissue may be difficult due to ethical reasons, time constraints, high costs or the overall low number of the target cell population. To overcome

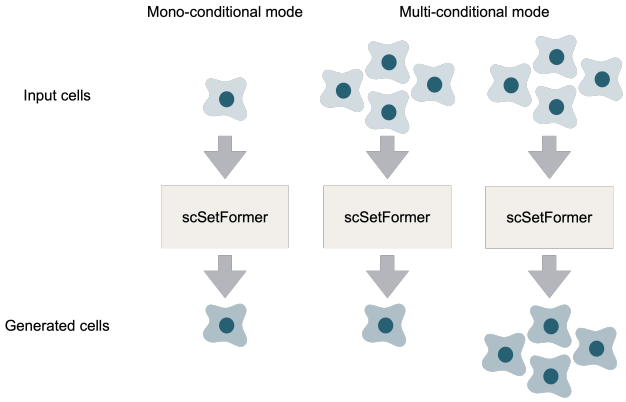

Figure 1. **Possible conditioning and generation modes of scSetFormer.** In the mono-conditional mode (left), scSetFormer can generate a cell conditioned on one input cell. In the multi-conditional mode (middle and right), scSetFormer can generate one cell (middle) or multiple cells (right) conditioned on an input set of multiple cells.

this problem, upsampling techniques have been applied to yield a larger number of the desired cell data (Marouf et al., 2020). However, this approach has been shown to potentially result in unreasonable conclusions (Marouf et al., 2020). The increasing success of generative machine learning models (Harshvardhan et al., 2020) has drawn the attention of scRNA-seq scientists from traditional *in vitro* towards the *in silico* generation of single cells (Lee, 2023) as more time and cost efficient approaches for performing scRNA-seq experiments and augmentation of small-sized cell population.

Generative models are capable of generating new yet realistic images (Khan et al., 2020), texts (Wu et al., 2023), or can be used for *de novo* drug design (Martinelli, 2022), based on the complex features they learn during their training process. Many models have been proposed for scRNA-seq generation based on such generative models (Harshvardhan et al., 2020). Variational autoencoders (Kingma & Welling, 2013), generative adversarial networks (Goodfellow et al., 2020), diffusion (Ho et al., 2020) and Transformer-based (Vaswani et al., 2017) models are the most common. While these models have different strengths and weaknesses, most of them also share limitations. First, the cells generated by some models tend to have less realistic gene expression profiles. Second, these models indirectly incorporate a particular order of genes into their framework, making the generation process incorrectly dependent on the order of the input genes. Third, all of these models rely on a single input cell to generate cells. While this may be desirable in some scenarios, the generation process lacks information about the expression profiles of other related cells.

In this work, we introduce a new scRNA-seq generative model—scSetFormer—which overcomes these limitations using a Set Transformer-based (Lee et al., 2019) variational autoencoder. scSetFormer leverages the attention mechanism to model dependencies across genes, within and across cells. The integration of set attention modules makes scSetFormer permutation-invariant, removing any false information that would render the generation process dependent on a particular order of the considered genes. The permutation invariance additionally eliminates any order bias and allow the model to solely focus on learning the dependencies across genes. Our model can, therefore, deal with different gene identities if trained on datasets with different gene order and identifiers. Furthermore, scSetFormer is also designed to generate cells conditioned on not only one but multiple input cells, which makes the generation process more informative due to the information aggregation across multiple cells. In dynamic scRNA-seq data, in which a cell is represented by its gene expression levels over different time steps t. An application for such data that can only be achieved by incorporating multiple input cells is the generation of a cell at time step t conditioned on cells from time steps $\{0, \dots, t-1\}$. All of these features make scSetFormer more flexible and render it a foundation for the conditional generation of scRNA-seq data for numerous scenarios.

2. Preliminaries

In this section we explain the data on which scSetFormer can be trained, i.e. scRNA-seq data. We also provide a brief explanation of each of its main building blocks.

2.1. scRNA-seq data

Information about gene expression levels obtained by single-cell RNA sequencing (scRNA-seq) is typically stored in a matrix M of size $n \times m$, where n refers to the number of cells detected and m to the number of genes sequenced. An element in M_{ij} refers to the expression level of gene j in cell i. A row M_i in this matrix is also called the *gene expression profile* of cell i, as it contains information about the expression levels of all sequenced genes.

The gene expression level is defined by the number of RNAs found in a cell, which are reverse transcribed into complementary DNA to determine the genes from which these RNAs were transcribed.

2.2. Transformer

Transformers (Vaswani et al., 2017) are designed to detect dependencies within long input data sequences, originally on language sequences. The main building block in Transformers is the self-attention mechanism, through which pairwise dependencies within the input data sequence, or the *attention* that each element in the input sequence has to each other element, are computed. The input data is encoded by the combination of both the word embeddings together with their positional embeddings. This is crucial for Transformers applied to language tasks, as the attention mechanisms are unaware of the positions of the sequence elements. To compute the self-attention scores, three linear projections Q, K and V are applied to the combined input embeddings. These projections are then used to compute the self-attention scores for the sequence elements via the scaled-dot product attention as follows:

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V,$$

where d_k is the dimension of the query and key vectors in Q and K respectively, which are multiplied to compute the attention weights. softmax refers to the softmax activation function used to normalise the attention weights scaled by $\sqrt{d_k}$. The value vectors in V are then multiplied by the

scaled attentions to produce the final attention scores. Another important building block that allows the detection of not only pairwise but also higher order dependencies in the input sequence is the multi-head attention (MHA) mechanism. In MHA, attention scores are computed h times, where h is the number of attention heads, each with its own weight W applied to the three linear projections. The final attention is the concatenation of all attentions from all attention heads:

$$\text{MHA}(Q, K, V) = \text{concatenate}(\text{head}_0, \ldots, \text{head}_h),$$

with

$$\text{head}_i = \text{attention}(QW_i^Q, KW_i^K, VW_i^V)W^O,$$

where $d_k = d_v = \frac{d_m}{h}$, $W_i^Q \in \mathbb{R}^{d_m \times d_k}$, $W_i^K \in \mathbb{R}^{d_m \times d_k}$ and $W_i^V \in \mathbb{R}^{d_m \times d_v}$ are attention weights of head i, and $W^O \in \mathbb{R}^{hd_v \times d_m}$ are the weights applied to the obtained attention scores of each head i.

Transformers further incorporate residual connection, batch normalisation, drop-out, and row-wise feedforward layers (Vaswani et al., 2017).

2.3. Permutation-invariant neural networks

Neural networks that learn from input data with set properties should be permutation invariant. Such networks consist of three parts:

1. **Encoder** ϕ: ϕ is a continuous function applied to each element of the input set to produce corresponding features, i.e. $\phi(\{x_0, \ldots, x_n\}) = \{f_0, \ldots, f_n\}$, where $\{x_0, \ldots, x_n\}$ are the elements of the input set $\mathcal{X}$.
2. **Pooling function** π: π must be a permutation invariant function which is used to aggregate the resulting features by the encoder, i.e. $\pi(\{f_0, \ldots, f_n\}) = p$. The pooling function can be the mean, sum, maximum, etc. This removes the need for the network to learn order dependant semantics, simplifying the learning process.
3. **Decoder** ψ: ψ is a continuous function which is applied to the aggregated feature vector p to predict the target output y, i.e. $\psi(p) = \hat{y}$.

The framework of a permutation invariant neural network for an input set $\mathcal{X} = \{x_0, \ldots, x_n\}$ with a target label y can therefore be summarised as follows:

$$\hat{y} = \psi(\pi(\phi(\{x_0, \ldots, x_n\}))).$$

2.4. Set Transformer

A Set Transformer (Lee et al., 2019) is a variant of the vanilla Transformers in which the positional encoding and the dropout layer are removed, making it suitable for predictions based on inputs with set properties, i.e. where the order of the elements in the input set has no effect on, and does not correlate with, the target output. Therefore, the input to the Set Transformer does not include the positional embeddings of the input sequence elements. The Set Transformer follows the architecture of a permutation invariant network. The building blocks that make up the encoder, decoder and pooling function are explained below.

Multi-head attention block The multi-Head attention block (MAB) is a Transformer layer with a MHA (Fig. 2). It computes the attention scores between the elements of two input sets $\mathcal{X}$ and $\mathcal{Y}$. The difference to a vanilla Transformer layer is the removal of the drop-out layer. MHB can be formulated as follows:

$$\begin{aligned}\text{MAB}(\mathcal{X}, \mathcal{Y}) &= \text{Norm}(\mathcal{H} + \text{rFF}(\mathcal{H})) \\ \mathcal{H} &= \text{Norm}(\mathcal{X} + \text{MHA}(\mathcal{X}, \mathcal{Y}, \mathcal{Y})),\end{aligned}$$

where Norm is the normalisation layer and rFF is the row-wise feedforward layer.

Set attention block The set attention block (SAB) is an MHB that acts as a self-attention to the elements of the input set $\mathcal{X}$ (Fig. 2). Multiple SABs are used to model higher order dependencies between the elements of the input set:

$$\text{SAB}(\mathcal{X}) = \text{MAB}(\mathcal{X}, \mathcal{X}).$$

Since it fulfils the properties of a continuous function, it can be used in both the encoder and decoder ϕ and ψ, respectively.

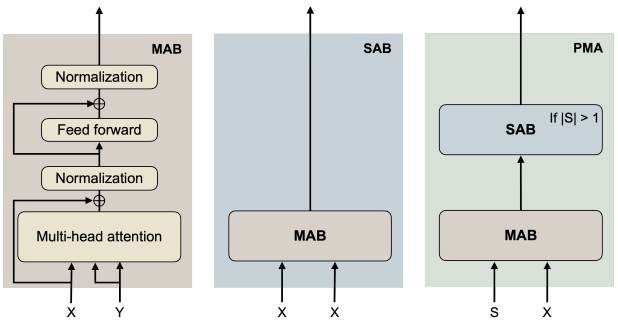

Figure 2. **Neural network blocks used in Set Transformers.** (a) Multi-head attention block (MAB). (b) Set attention block (SAB). (c) Pooling by multi-head attention (PMA). X and Y are input sets and S is a set of seed vectors.

Pooling by multi-head attention As a pooling operation, the Set Transformer applies an MHB to its input vectors $\mathcal{Z}$, which are reduced to a set $\mathcal{S}$ of size $k = 1$, while the resulting vector corresponds to the aggregated vector p (Fig. 2).

$$\text{PMA}_k(\mathcal{Z}) = \text{MAB}(\mathcal{S},\ \text{rFF}(\mathcal{Z})).$$

If $k > 1$, (Lee et al., 2019) apply SAB to $\mathcal{Z}$ to compute further dependencies between the aggregated vectors:

$$\text{SAB}(\text{PMA}_k(\mathcal{Z})).$$

2.5. Variational autoencoders

Variational autoencoders (VAEs, Kingma & Welling 2013) are probabilistic generative models that consist of two parts: an encoder and a decoder. The main function of the encoder is to reduce the input data from its high dimensional space to a lower dimensional space called the latent space. This reduction helps to track the main features in the input space, ignoring potential noise. It approximates a Gaussian variational posterior distribution $q(z|x)$, which is at the core of a VAE's ability to generate new data. Furthermore, a latent vector z is sampled from $q(z|x)$, which is then passed to the decoder to reconstruct the input data x.

VAEs are generally optimised by the evidence lower bound objective (ELBO), which is defined as:

$$\begin{aligned}\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\theta(z|x)}[&\log p_\phi(x|z) \\ &- \log q_\theta(z|x) \\ &+ \log p(z)] \leq \log p_\phi(x),\end{aligned}$$

where θ and ϕ are the parameters for the encoder $q_\theta(z|x)$ and decoder $p_\phi(x|z)$, respectively. $p(z)$ refers to the prior over the latent variables, typically assumed to be an isotropic Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

3. scSetFormer

The general conditional generation of scRNA-seq data framework in scSetFormer consists of mainly three steps:

- Generate a cell embedding E based on the gene names and gene expression values of an input cell x.
- Approximate the variational posterior distribution in the latent space $q_\theta(z|x)$ via the encoder.
- Reconstruct the gene expression profile $\hat{x}$ of the input cell x via the decoder after sampling a latent vector z from the approximated latent space posterior distribution.

scSetFormer allows the generation of new cells conditioned on one input cell as well as on a set of input cells. The generation of cells conditioned on one cell is denoted as *mono*-conditional mode (Fig. 1, left), while the generation of new cells conditioned on multiple cells is denoted as *multi*-conditional mode (Fig. 1, right). These two modes introduce two architectures for scSetFormer, which differ slightly. The differences and similarities will be highlighted in the following subsections.

3.1. Cell embedding

Several methods have been proposed to embed gene names $\mathcal{G}$ and gene expression profiles $\mathcal{P}$. Here we use an auto-discretisation method introduced in (Gong et al., 2024). In this method, gene names in $\mathcal{G}$ are first hashed into integers, followed by projection, resulting in an embedding $E_\mathcal{G} \in \mathbb{R}^{g \times d}$, where $g = |\mathcal{G}|$ and d is the embedding dimension. The corresponding gene expression levels in $\mathcal{P}$ of an input cell are embedded via a sequence of projection layers, resulting in a smooth binning of continuous gene expression values. Furthermore, the embedding $E_\mathcal{P}$ of a profile $\mathcal{P}$ is determined as follows:

$$\begin{aligned}v_1 &= w_1(\mathcal{P}) \\ v_2 &= \text{leaky_relu}(v_1) \\ v_3 &= w_2(v2) + \alpha \times v_2 \\ v_4 &= \text{softmax}(v_3) \\ E_\mathcal{P} &= v_4 \times \mathbf{T},\end{aligned}$$

where the vectors w_1 and w_2 and the scalar α are learnable weights. $\mathbf{T}$ is a lookup table of size $d \times t$, where t is the number of tokens. $E_\mathcal{G}$ and $E_\mathcal{P}$ are summed pairwise, resulting in $E \in \mathbb{R}^{g \times d}$, which represents the final embedding of an input cell:

$$E = E_\mathcal{G} + E_\mathcal{P}.$$

The cell embedding method is identical for both mono- and multi-conditional modes.

3.2. Encoder

Mono-conditional mode In this mode, the encoder first processes the input cell embedding E into a hidden layer so that the dimension of E matches the expected dimension of the self-attention heads. This is then passed to four multi-head set attention blocks, which model the dependencies across the genes of the input cell (Fig. 3). The resulting attentions are fed into four multi-head attention pooling blocks, resulting in a vector $p \in \mathbb{R}^{d_p}$. Finally, p is used to compute the parameters of the approximated latent space distribution $q_\theta(z|x)$:

$$q_\theta(z|x) = \mathcal{N}(\mu(p),\ \text{diag}(\sigma(p)^2)),$$

where θ refers to the learnable weights of the encoder and $z \in \mathbb{R}^{d_z}$ refers to a latent vector. Finally, a latent vector z is sampled from $q_\theta(z|x)$ and passed to the decoder.

Multi-conditional mode As the number of cells in this mode is greater than 1, the encoder here is designed to model both the dependencies of gene expressions *within* each cell and *across* cells (Fig. 4). The former is achieved by using four set-attention blocks with common weights for

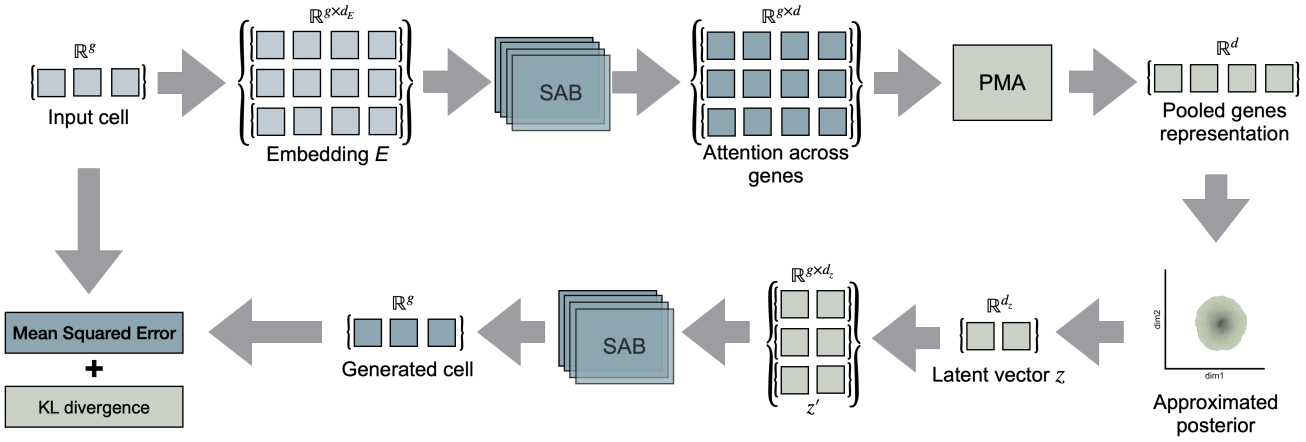

Figure 3. **scSetFormer architecture in its mono-conditional mode.** The loss function of scSetFormer in this mode comprises the reconstruction loss and the Kl-divergence between the latent prior and approximated posterior distributions.

each input cell in the input set, while the latter is achieved by applying four additional set-attention blocks to the resulting attentions of set-attention blocks modelling the *across* dependencies. Similarly to the mono-conditional mode, the attentions from the final steps are passed to a pooling block to compute the parameters of $q_\theta(z|x)$.

3.3. Decoder

Mono-conditional mode In this generation mode, the decoder has a similar architecture to the encoder. Its input latent vector z is passed into a dense layer $\mathbb{R}^{d \times g}$, so that after reshaping, z adapts to a shape of $\mathbb{R}^{g \times d}$. This step is crucial as it allows the reconstruction of a cell consisting of g genes, equal to the number of genes in the input cell. The processed latent vector z is then fed into four multi-headed attention blocks. The resulting attentions are passed into a final dense layer, resulting in $\hat{x} \in \mathbb{R}_+^g$, which corresponds to the reconstructed gene expression profile of the input cell x.

Multi-conditional mode The decoder in this mode has a similar architecture to that of the mono-conditional mode, with an additional dense layer. Similar to the dense layer in the mono-conditional mode decoder, which is used to ensure the reconstruction of a cell with g genes, this additional dense layer adds an extra dimension to the latent vector of size m, corresponding to the number of cells in the reconstructed set $\hat{\mathcal{X}}$.

The number of all set attention blocks is a hyperparameter determined during the performance optimisation of scSetFormer.

3.4. Optimisation objective

Artificial neural networks learn the desired behaviour by optimising an objective function, also called the *loss* function. The loss function gives a measure of the deviation from the desired behaviour, or the loss, that the network would produce based on the current set of learnable parameters. scSetFormer is trained, as in vanilla VAEs, to optimise the evidence lower bound (ELBO):

$$\begin{aligned}\mathcal{L}(x;\theta;\phi) = \mathbb{E}_{q_\theta(z|x)}[\log p_\phi(x|z)] \\ - \beta \cdot D_{\mathbf{KL}}[q_\theta(z|x)\|p(z)],\end{aligned}$$

where $\mathbb{E}_{q_\theta(z|x)}[\log p_\phi(x|z)]$ is the reconstruction loss between the input cell x and the reconstructed cell $\hat{x} \sim q_\theta(z|x)$, $p_\phi(x|z)$ is the decoder, parameterised by learnable weights, ϕ, $p(z)$ is the latent prior, chosen to be an isotropic Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and $D_{\mathbf{KL}}$ is the Kullback-Leibler divergence between the latent approximated posterior $q_\theta(z|x)$ learned by the encoder and the latent prior $p(z)$. The optimisation objective in both modes differs in the choice of the reconstruction loss.

Mono-conditional mode In this mode, we use the mean squared error loss as the reconstruction loss. It is defined as follows:

$$\text{MSE} = \frac{1}{b}\sum_{i=1}^{b}(\hat{x}_i - x_i)^2,$$

where b is the size of one input batch.

Multi-conditional mode Since in this mode the input to the model is a set of cells, and potentially a set of cells as output, a loss reflecting the distance between all cells in the input set $\mathcal{X}$ and all cells in the generated set $\hat{\mathcal{X}}$ must

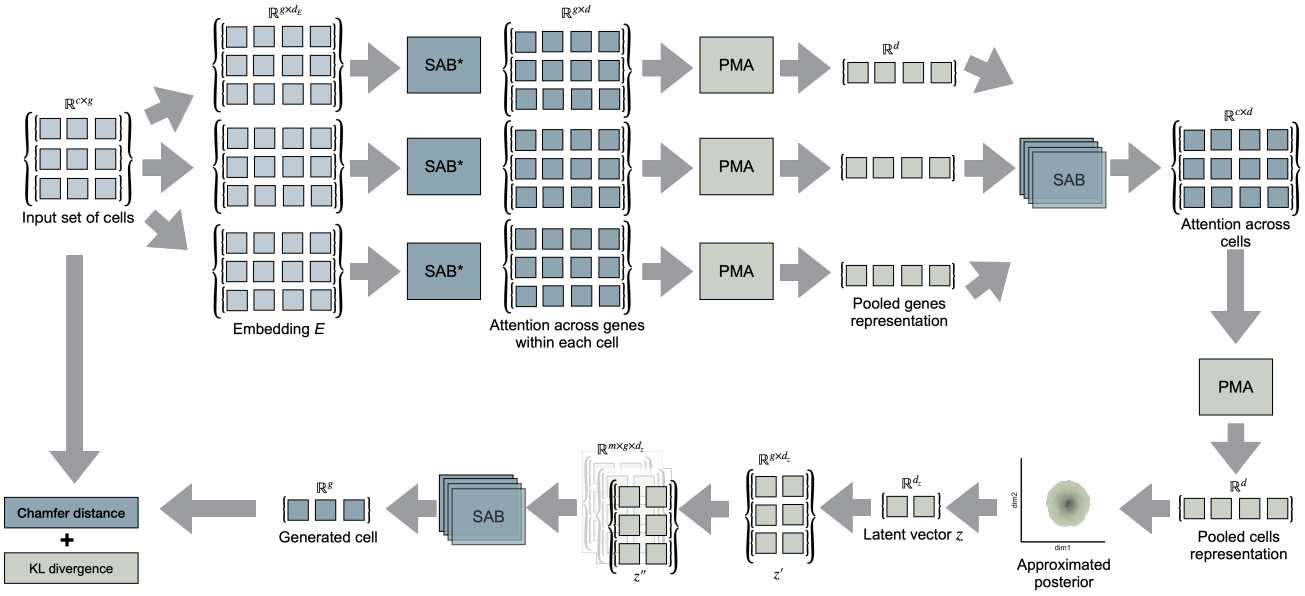

Figure 4. **scSetFormer architecture in its multi-conditional mode.** The loss function of scSetFormer in this mode comprises the Chamfer distance and the KL-divergence between the latent prior and approximated posterior distributions. * indicates weight sharing.

be considered. Here we use the Chamfer distance (Barrow et al., 1978), which is designed to compute the distance between two sets. Using this part of the loss, the averaged sum of the squared shortest distance from each cell in $\mathcal{X}$ to its nearest cell in $\hat{\mathcal{X}}$ and vice versa is minimised:

$$\text{Chamfer}(\hat{\mathcal{X}}, \mathcal{X}) = \frac{1}{|\hat{\mathcal{X}}|} \sum_{\hat{x} \in \hat{\mathcal{X}}} \min_{x \in \mathcal{X}} \text{Huber}(\hat{x}, x) + \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \min_{\hat{x} \in \hat{\mathcal{X}}} \text{Huber}(x, \hat{x}),$$

where the Huber loss (Gokcesu & Gokcesu, 2021) is used to determine the squared distances between x and $\hat{x}$. It aims to combine the properties of both the mean squared error and the mean absolute error:

$$\text{Huber}(x, \hat{x}; \delta) = \begin{cases} \frac{1}{2}(x - \hat{x})^2, & \text{if } |x - \hat{x}| \leq \delta, \\ \delta |x - \hat{x}| - \frac{1}{2}\delta^2, & \text{otherwise,} \end{cases}$$

where δ is a threshold parameter that defines the transition between squared loss and absolute loss. Here it is set to its default value of 1.

4. Experiments

Our experiments are designed to evaluate the quality of cells generated by scSetFormer compared to other scRNA-seq data generative models using four statistical distances. Additionally, we analyse the quality of scSetFormer's latent space, which is at the heart of the generation process in VAEs. Furthermore, we perform four experiments to uncover any memorisation-based generation. Finally, we perform three most common scRNA-seq analyses to assess the underlying biology of the generated cells. In general, we primarily analyse the mono-conditional mode of scSetFormer, leaving a more comprehensive evaluation of the multi-conditional mode for future work.

4.1. Datasets

We trained all models on two datasets containing human peripheral blood mononuclear cells (PBMCs, Yazar et al. 2022) and mouse neurons Zheng et al. (2017). An overview of the two datasets can be found in Table 1. The neurons

Table 1. **Overview of the PBMCs by Yazar et al. (2022) and neurons Zheng et al. (2017) datasets.** The number of used cells indicates the number of cells used to train and evaluate the models.

Dataset	#Total cells	#Cell types	#Used cells
PBMCs	1,248,980	31	108,000
Neurons	20,000	None	20,000

dataset has already been processed. The PBMCs dataset was processed following the standard scRNA-seq preprocessing framework using the Python package scanpy (Wolf et al., 2018): cells expressing less than three genes and cells with total gene expression counts greater than 15,000 were filtered out. Gene expression profiles are normalised to a target sum of 10,000 counts, followed by a $\log_{1p}$ transformation. In all experiments, we use the top 100 highly variable genes

determined using scanpy. Due to the long training time of scSetFormer caused by the attention score computation, we select a smaller subset of the PBMCs dataset (Tab. 1), all contained cells are from four different cell types, namely: B memory, CD4$^+$ CTL, CD8$^+$ TCM and natural killer T cells. The choice of these cell types is intended to test the ability of the model to generate cells that have similar gene expression profiles but belong to different cell types, i.e. the three T cell types.

4.2. Conditional generation

scSetFormer aims at the conditional generation of scRNA-seq data. We divide the conditional generation process into two main modes depending on the number of input cells: *mono-* and *multi*-conditional modes (Fig. 1).

Mono-conditional mode In mono-conditional mode, scSetFormer generates a single cell conditioned on a single input cell as follows:

$$\begin{aligned} q_\theta(z|x) &= \mathcal{N}(\mu(x),\ \text{diag}(\sigma(x)^2)) \\ z &\sim q_\theta(z|x) \\ \hat{x} &\sim p_\phi(x|z), \end{aligned}$$

where x corresponds to the input and $\hat{x}$ to the generated cell, and z is the latent vector that is sampled from the approximated variational posterior distribution $q_\theta(z|x)$ conditioned on the input cell x. Sampled negative expression values are clipped to zero as a correction method.

Multi-conditional mode The multi-conditional mode incorporates information about multiple cells at once during the generation process, so the size of the input set of cells $\mathcal{X}$ here is n, where n is greater than 1. This mode is divided into two further categories: *mono-*$\hat{\mathcal{X}}$ and *multi-*$\hat{\mathcal{X}}$, reflecting the number of generated cells. In the case of mono-$\hat{\mathcal{X}}$, scSetFormer generates a set with a single cell $|\hat{\mathcal{X}}| = 1$:

$$\begin{aligned} q_\theta(z|\mathcal{X}) &= \mathcal{N}(\mu(\mathcal{X}),\ \text{diag}(\sigma(\mathcal{X})^2)), |\mathcal{X}| = n \\ z &\sim q_\theta(z|\mathcal{X}) \\ \hat{\mathcal{X}} &\sim p_\phi(\mathcal{X}|z), |\hat{\mathcal{X}}| = 1. \end{aligned}$$

In the case of multi-$\hat{\mathcal{X}}$, scSetFormer generates a set of cells that either has a size greater than 1 or the same size as the input set $\mathcal{X}$, i.e. $|\hat{\mathcal{X}}| = |\mathcal{X}| = n$:

$$\begin{aligned} q_\theta(z|\mathcal{X}) &= \mathcal{N}(\mu(\mathcal{X}),\ \text{diag}(\sigma(\mathcal{X})^2)), |\mathcal{X}| = n \\ z &\sim q_\theta(z|\mathcal{X}) \\ \hat{\mathcal{X}} &\sim p_\phi(\mathcal{X}|z), |\hat{\mathcal{X}}| = n. \end{aligned}$$

Conditional generation on a set of multiple cells, together with the respective generated cells, gives scSetFormer great flexibility to be applied to different scRNA-seq generation tasks with different goals. In this work, we train the model on an input set of size 4, where all cells in a set are of the same cell type. Furthermore, we focus on the evaluation of the mono-conditional mode and leave other possible optimisation and applications of scSetFormer to future work.

4.3. Unconditional generation

In our work, we also investigate the unconditional generation of new gene expression profiles x', i.e. without conditioning the input cell. This is done by sampling a latent

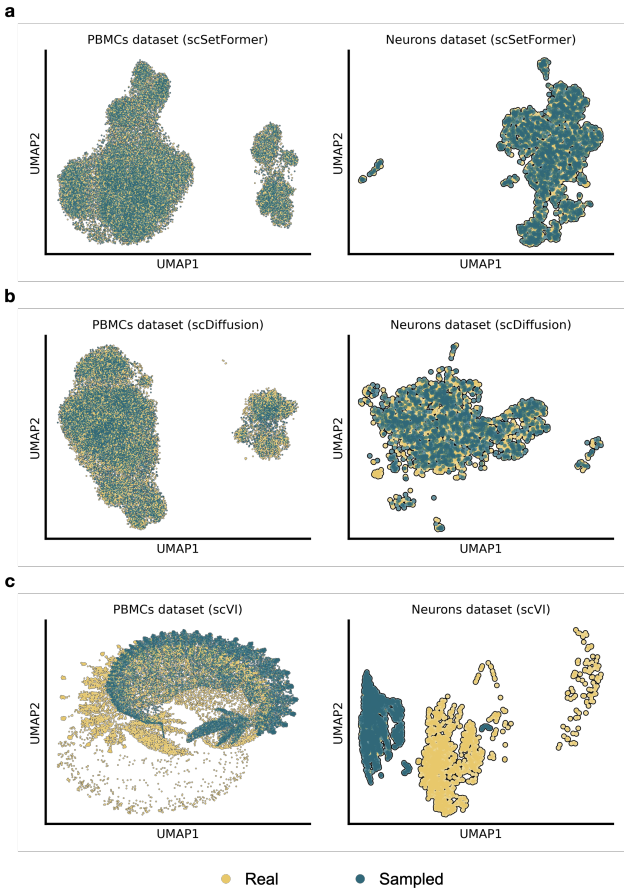

Figure 5. **Real and sampled cells projected via UMAP.** PBMCs data is shown in the left and the neurons data in the right column. Real cells and sampled cells by scSetFormer (a), scDiffusion (b) and scVI (c).

vector z from the latent prior distribution and passing it to the decoder $p(x'|z)$ to generate a new cell. In this experiment, we encountered the *posterior collapse* problem, where we observe that the unconditionally generated cells have different gene expression profiles than the real cells (Fig. A5). We therefore looked for alternative priors to mitigate this problem. We use two flexible priors as alternatives to the isotropic Gaussian: a Gaussian mixture model and

a flow-based model. We apply scSetFormer with the flexible priors to the PBMCs dataset and to a toy dataset to better understand their behaviour. To learn more about the conclusions after using flexible, we refer to the Appendix A2.

4.4. Model comparison

As the multi-conditional mode is unique to scSetFormer, we only compare the mono-conditional mode with two generative models from different backgrounds: the variational inference-based model scVI (Gayoso et al., 2022) and a recently published diffusion-based generative model scDiffusion (Luo et al., 2024). Further details of their underlying frameworks can be found in section 5. For both models we use their default settings in all respective experiments.

4.5. Projection via UMAP

Uniform manifold approximation and projection for dimension reduction (UMAP, McInnes et al. 2018) is widely used in scRNA-seq visualisation as it provides insight into the spatial arrangement of cells in low dimensional feature space. To test whether the generated gene expression profiles show a overlapping spatial arrangement to the real cells, we compute a UMAP for the real and generated cells by each generative model, separately.

Cells generated by scSetFormer and scDiffusion for both datasets show a high spatial similarity and overlapping to the real cells (Fig. 5a, b). The different appearance of the cell arrangement of both real and scVI-generated cells (Fig. 5c) can be explained by the fact that scVI is trained on raw count data, i.e. not normalised or log-transformed, in addition to the small number of genes included in this work (100 genes), which could have led to the loss of clustering arrangements. Nevertheless, the cells generated by scVI show less overlap with the real cells.

4.6. Benchmarking metrics

In generative models, the quality of the generated data is typically evaluated by comparing the underlying distributions of the real and generated datasets. Here, we use three different and widely used distances to evaluate the underlying distribution of the generated cells: the maximum mean discrepancy (MMD, Gretton et al. 2012), the sliced Wasserstein distance (SWD, Bonneel et al. 2015) and the classifier two-sample test (C2ST, Lopez-Paz & Oquab 2016). Additionally, we use the Silhouette score (Shahapure & Nicholas, 2020) to measure how well the generated cells fit their target cluster cell type.

Maximum mean discrepancy (MMD) MMD is a distance which compares the mean difference of the data points after being mapped onto a feature space. The general formula is defined as follows:

$$\text{MMD}^2[\phi, P, Q] = \|E_{x\sim P}[\phi(x)] - E_{\hat{x}\sim Q}[\phi(\hat{x})]\|^2,$$

where ϕ corresponds to the feature map and P and D correspond to the underlying distribution of the real and generated data, respectively. The feature maps are computed using the *kernel trick* (Schölkopf, 2000), where the feature embeddings of the data points in the feature space are not computed explicitly, but a kernel function k is applied to pairs of data points. Here we use an MMD with a Gaussian kernel:

$$k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2},$$

where γ is a hyperparameter that controls the bandwidth of the Gaussian kernel, and the lower the MMD, the greater the overlap between the real and generated datasets. In our experiments, we observed inconsistencies when using MMD as a quality criterion, as discussed in the Appendix. Therefore, we adjusted the gamma from the default to $\gamma = 8.0e-05$ for the PBMCs and $2.8e-04$ for the neurons dataset. See Appendix A1 for more details.

With respect to the MMD, scSetFormer outperforms scDiffusion and scVI on the PBMCs dataset (Tab. 2). On the neurons dataset, scDiffusion outperforms scSetFormer and scVI (Tab. 3). scSetFormer and scDiffusion differ, however, in a very small margin on both datasets.

Sliced Wasserstein distance (SWD) SWD is a metric based on the earth mover distance (Andoni et al., 2008), which aims at estimating the minimum cost of optimal transport from the probability distribution P to the probability distribution Q. Other variants of the earth mover distance have been integrated into the optimisation objectives of both GANs (Gulrajani et al., 2017) and VAEs (Kolouri et al., 2018), due to their robustness and high coverage of multi-model distributions. SWD allows a tractable and efficient computation of the optimal transport by linearly projecting the data points onto predefined vectors p and q. In this work, we follow (Kolouri et al., 2018) to compute an approximation of the SWD, where the projected data are then sorted, and the mean absolute difference between the sorted projections defines the SWD:

$$\text{SWD}(P, Q) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\left|p_{\theta_i,j} - q_{\theta_i,j}\right|,$$

where $p_{\theta_i,j}$ and $q_{\theta_i,j}$ are the j-th elements of the sorted projections of samples from P and Q, respectively, onto the i-th direction, and m is the number of samples in each projection.

Table 2. **Benchmarking metrics on the PBMCs dataset.** MMD: maximum mean discrepancy, SWD: sliced-Wasserstein distance, C2ST: classifier two-sample test, $\overline{\rho}$: mean nearest neighbour ratio, m: slope of a linear fit. Best values are bold and second best values are underlined.

Model	MMD	SWD	C2ST	C2ST+PCA	$\overline{\rho}$	m
Baseline	7.868e-06	0.044	0.501	0.501	-	-
scVI	5.603e-05	1.024	**0.604**	<u>0.568</u>	infinity	0.563
scDiffusion	<u>6.676e-06</u>	**0.278**	<u>0.665</u>	0.599	**1.001**	**0.904**
scSetFormer	**5.722e-06**	<u>0.331</u>	1.0	**0.498**	<u>1.03</u>	<u>0.839</u>

Table 3. **Benchmarking metrics on the neurons dataset.** MMD: maximum mean discrepancy, SWD: sliced-Wasserstein distance, C2ST: classifier two-sample test, $\overline{\rho}$: mean nearest neighbour ratio, m: slope of a linear fit. Best values are bold and second best values are underlined.

Model	MMD	SWD	C2ST	C2ST+PCA	$\overline{\rho}$	m
Baseline	1.311e-06	0.087	0.495	0.507	-	-
scVI	2.62e-04	2.409	**0.563**	**0.511**	infinity	0.764
scDiffusion	**2.337e-05**	**0.267**	<u>0.569</u>	<u>0.551</u>	<u>1.139</u>	**0.866**
scSetFormer	<u>4.708e-05</u>	<u>0.308</u>	0.999	0.303	**1.073**	<u>0.84</u>

In this metric, scDiffusion outperforms all models on both PBMCs and neurons datasets (Tab. 2, 3). scSetFormer differs from scDiffusion by a margin of about 0.05 on both datasets, while it outperforms scVI by a large margin of 0.7 and 1.1 on the PBMCs and neurons datasets, respectively.

Classifier two-sample test (C2ST) C2ST is a classifier-based distance where a binary classifier is trained to distinguish between the underlying distributions of two datasets. Here we use a random forest classifier (Breiman, 2001) consisting of 1000 trees implemented by scikit-learn (Kramer & Kramer, 2016). The accuracy of C2ST is used to evaluate the quality of the generated cells. An accuracy of 50% would mean that the two distributions are indistinguishable, an accuracy of 100% would mean that they are clearly distinguishable. C2ST therefore provides a highly interpretable measurement.

C2ST is highly sensitive to outliers when applied to high dimensional data (Bischoff et al., 2024). We therefore evaluate the accuracy of traditional C2ST by the accuracy obtained after applying C2ST to the top principal components that explain 95% of the data variance. We call this measure C2ST+PCA. When the C2ST is applied to the raw data, scVI and scDiffusion outperform scSetFormer (Tab. 2, 3). scSetFormer leads to an accuracy of approximately 1.0 on both datasets. When the C2ST is applied to the top PCs explaining 95% of the data variance, scSetFormer outperforms both models with a C2ST accuracy of 0.498 on the PBMCs (Tab. 2). On the neurons dataset, scSetFormer results in an accuracy that worse than a random guess (Tab. 3). The receiver operator characteristic (ROC) curves are shown in Fig. 7.

To gain a better understanding of the large discrepancy in the C2ST performance on the raw data generated by scSetFormer, we examine the genes with the highest importance factor for the prediction of the random forest classifier. We determined that the generated expression values of the gene with the highest importance factor are greater than zero for most of the cells, despite having zero expression value in most of the real cells (Fig. A15a). The perfect accuracy of the C2ST on the raw data stems, therefore, from the classifier making decisions based on such outlier genes. However, when these are removed by the PCA, the C2ST is no longer able to distinguish between the real and generated cells.

We notice that the C2ST+PCA applied to the neurons data generated by scSetFormer results in an even worse performance than a random classifier (Fig. 7b, bottom). We hypothesise that this may be due the amount of variance removed when only 95% of the data variance is considered. We, therefore, examine the performance of C2ST+PCA by considering 96%, 97%, 98% and 99% of the data variance. We can see that adding more variance leads to a gradually better performance (Fig. A15b), suggesting that using only 95% of the data variance on this dataset discards relevant information that are vital for the separation, leading to a worse performance than random guessing.

Silhouette score To test how well a generated cell $\hat{x}$, conditioned on an input cell x of type t, fits the cluster of cells

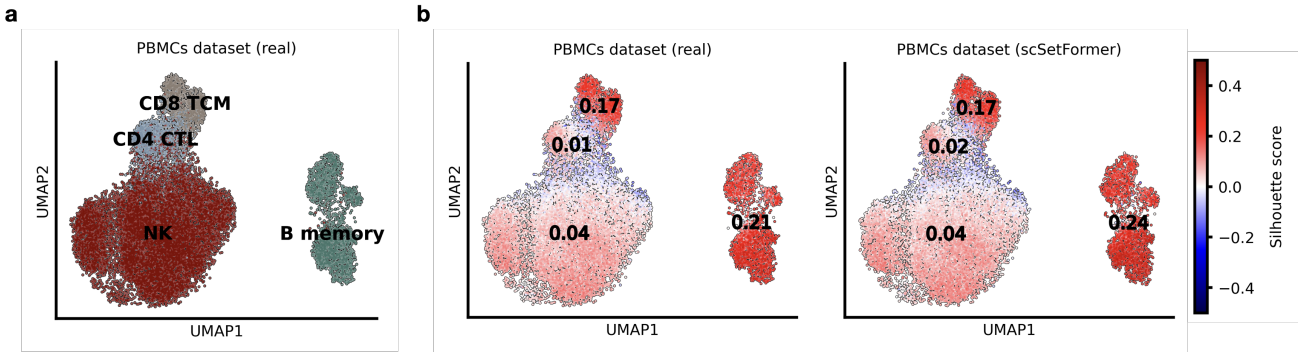

Figure 6. **Cluster quality in the PBMCs dataset.** (a) Real cells of the test dataset coloured by their cell type, shown as a reference. (a) Real and sampled cells coloured by their Silhouette score. The average of Silhouette score for each cell cluster is shown on the corresponding cell cluster.

of type t, we use the Silhouette score, defined as follows:

$$s(\hat{x}) = \frac{b - a}{\max(a, b)},$$

where a is the average intra-cluster distance, defined as the average distance between $\hat{x}$ and all other cells of type t, and b is the average nearest cluster distance, defined as the smallest average distance from $\hat{x}$ to all cells in any other cluster, not including clusters of cells of type t to which $\hat{x}$ belongs.

The Silhouette score ranges from -1 to 1, where:

- $s = +1$ indicates that $\hat{x}$ resembles well to its own cluster t, and poorly resembles neighbouring clusters.
- $s = 0$ indicates that $\hat{x}$ is very close to the decision boundary between two neighbouring clusters.
- $s = -1$ indicates that $\hat{x}$ poorly resembles its own cluster t.

We calculate the Silhouette score for each real and generated cell. The average Silhouette score across all clusters could be misleading, as the three adjacent clusters, i.e. NK, CD8 TCM and CD4 CTL cells, would produce small Silhouette scores that are difficult to compare and interpret overall. We therefore compare the average Silhouette score within each cluster.

scSetFormer generates cells with Silhouette scores highly similar to the real cells (Fig. 6b, A11d), indicating a high cluster quality of the generated cells. Similar to the real cells, some generated cells show negative Silhouette scores as they lie between two clusters of different cell types. B memory cells have the highest average Silhouette score, as they are far away from the other three T cell clusters.

4.7. Memorisation criteria

Conditional generation using generative models such as VAE are prone to learning the identity function, i.e. $\hat{x} = x$, rather than learning to generate $\hat{x}$ distinct from, but similar to x (Bozkurt et al., 2019). To judge whether scSetFormer suffers from this problem, we use three criteria.

Variance in latent space The first criterion is to examine the variance in the latent space. Variance close to zero indicates learning of the identity function, as in this case the model learns an almost exact mapping from input space to latent space with no variance, as opposed to variance greater than zero.

scSetFormer passes this check as the variance in all latent space dimensions is close to 1.0 (Fig. 8b, A11c). In general, this is a result of the KL divergence term being included in the optimisation objective. The KL divergence forces the approximated posterior distribution to conform to an isotropic Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, i.e. the latent prior, which has a variance of 1.0, while the reconstruction loss competes with the KL divergence to generate new cells similar to the input cells. Depending on how large β is, which controls the contribution of the KL divergence to the final loss, VAEs can theoretically learn the identity function when $\beta \approx 0$. In this case, VAEs are rendered traditional autoencoders, where the latent vectors correspond to the mean of the approximated posterior distribution.

Multiple generation For the second criterion, we pass a cell x to scSetFormer 10 times to generate cells conditioned on x. We then examine the gene expression profiles of the 5 generated cells. Then the gene expression value of gene g in x, i.e. its ground truth expression value, is compared with its expression value in the 10 generated cells $\hat{x}_i$. Equal expression values for gene g in all generated cells indicate

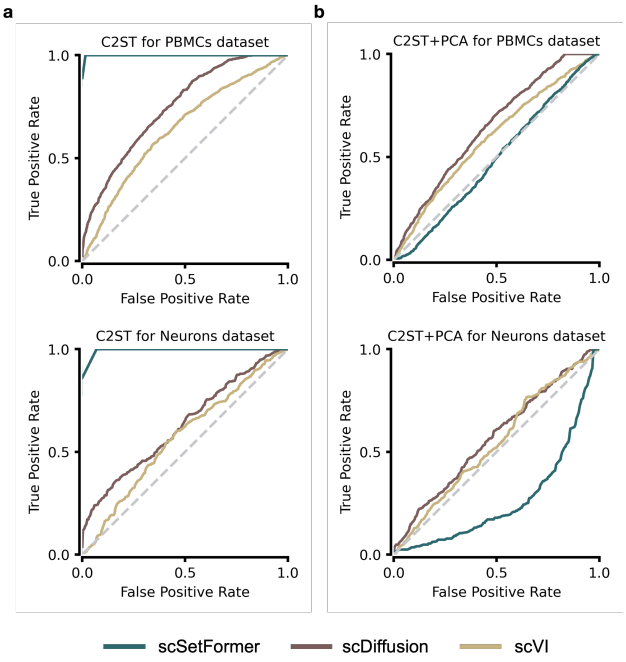

Figure 7. **Receiver operator characteristic (ROC) curve of the C2ST.** The PBMCs dataset is shown in the top and the neurons dataset in the bottom row. ROC curves of the C2ST applied to the raw data (a) and to the top principal components (PCs) explaining 95% of the data variance (b).

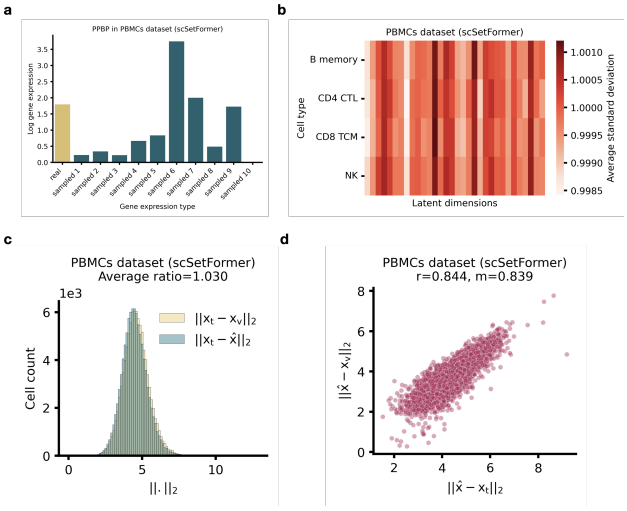

Figure 8. **Memorisation checks of scSetFormer on the PBMCs dataset**. (a) Real and sampled gene expression values for the gene PPBP in a cell. (b) Heatmap showing the standard deviation for each latent dimension averaged over latent vector clusters determined by the cell type of the corresponding input cells. (c) Distribution of the Euclidean distance between the real cells in the training dataset x_t and the real cells in the validation dataset x_v (yellow) and the Euclidean distances between the training dataset and the generated cells $\hat{x}$ (blue). (d) Euclidean distances of the generated cells to the training dataset versus Euclidean distances of the generated cells to the validation dataset. r refers to the Pearson correlation coefficient and m refers to the slope of a linear fit.

that the model is learning the identity function. If they are different, it is because a different latent vector was sampled each time to generate an output cell, which is a result of non-zero variance in the latent space.

scSetFormer generates different gene expression values in each generated cell (Fig. 8a). Therefore, we exclude the possibility that scSetFormer memorises the input cells or the training dataset.

Memorisation score The final check is to use a memorisation score, which aims to measure how much the model memorises, rather than generating new meaningful samples. Here we use the mean nearest neighbour ratio $\overline{\rho}$ (van den Burg & Williams, 2021). It measures the mean ratio of the Euclidean distance between a training cell $x_t \in \mathcal{X}_t$ to its nearest neighbour x_v from the validation set $\mathcal{X}_v$ and to its nearest neighbour $\hat{x}$ from the generated set $\mathcal{X}_g$:

$$\overline{\rho} = \frac{1}{n} \sum_{x_t \in \mathcal{X}_t}^{n} \frac{\min_{x_v \in \mathcal{X}_v} \|x_t - x_v\|_2}{\min_{\hat{x} \in \mathcal{X}_g} \|x_t - \hat{x}\|_2}.$$

$\overline{\rho}$ ranges from 0 to ∞, where:

- $\overline{\rho} \gg 1.0$ indicates that the generated cells are on average less similar to the training cells than the nearest neighbours of the validation cells, indicating poor quality generation.
- $\overline{\rho} \approx 1.0$ indicates that the generated cells are on average more similar to the conditioned cells than the nearest neighbours of the training cells from the validation cells, indicating high quality generation.
- $\overline{\rho} \ll 1.0$ indicates that the generated cells are on average more similar to the cells being conditioned on than the nearest neighbours of the training cells from the validation cells, indicating cell memorisation rather than new cell generation.

We further use the slope m of a linear fit between the $\|\hat{x} - x_t\|_2$ and $\|\hat{x} - x_v\|_2$ as a measure of the memorisation-based generation, where $m \approx 1.0$ indicates no data memorisation. $m = 1.0$, however, cannot be achieved as the cells in the training and validation datasets are different.

Our results show that scSetFormer and scDiffusion do not memorise the data on the PBMCs dataset, indicated by $\overline{\rho}$ and m close to 1.0 (Fig. 8c, d, A14). scDiffusion, however, results in a $\overline{\rho}$ larger than that of scSetFormer

on the neurons dataset (A12f, Fig. A13). scVI results in an infinite average ratio $\overline{\rho}$ and a low m on both datasets (Fig. A13a , A14a), indicating data memorisation.

4.8. Latent space

scSetFormer is a generative model based on the architecture of VAEs. Its latent space is at the heart of the reconstruction and generation of new samples, and thus plays a crucial role in the success of the generative process. We therefore evaluate whether the latent space preserves meaningful features and whether it is interpolatable.

Feature preservation in latent space In VAEs, the latent vector is usually a vector of lower dimension than the input vector. It's designed to capture the most relevant features in the input space, which are then used to generate the target data with high quality (Kingma & Welling, 2013). To evaluate whether the learned latent space in scSetFormer preserves biologically meaningful features that make up the distinct cell type, we visualise the latent vectors $z \sim q(z|x)$ for all input cells x of type t_x using UMAP. We then colour the latent vectors according to the cell type of the corresponding input cell t_x.

Our results show a high quality clustering of the latent vectors (Fig. 9a, A11b, A12b), although the latent vectors had no access to the labels of the input cells, i.e. the latent vectors are clustered in an unsupervised manner. Thus, scSetFormer is able to retain biologically meaningful representations of the input cells, making its latent space suitable for unsupervised clustering tasks.

Interpolation in latent space The latent space of VAEs should be smooth, i.e. small changes in the latent vector cause small changes in the generated data. This has various benefits on the model's generalisation, the interpretation of individual latent dimension, and the interpolation and thus generation of new unseen data (Berthelot et al., 2018). To examine the smoothness of scSetFormer's latent space, we interpolate between an NK T cell $\mathbf{z}_{NK}$ and an CD8 TCM cell $\mathbf{z}_{CD8}$ over 50 steps. We apply an interpolation factor $\alpha_i \in [0, 1]$ to resemble the interpolated vectors $\mathbf{z}_i$ between the two latent vectors:

$$\mathbf{z}_i = (1 - \alpha_i) \cdot \mathbf{z}_{NK} + \alpha_i \cdot \mathbf{z}_{CD8}, \; \forall \, i \in \{0, \ldots, 10\},$$

$\mathbf{z}_i$ is passed to the decoder to generate the corresponding cell. For $\alpha_i = 0$ scSetFormer would generate an NK T cell, and for $\alpha_i = 1$ scSetFormer would generate a CD8 TCM cell. For the in-between α_i scSetFormer would generate new cells that may not have been seen during training or may reflect the transition from an NK T cell to a CD8 TCM cell.

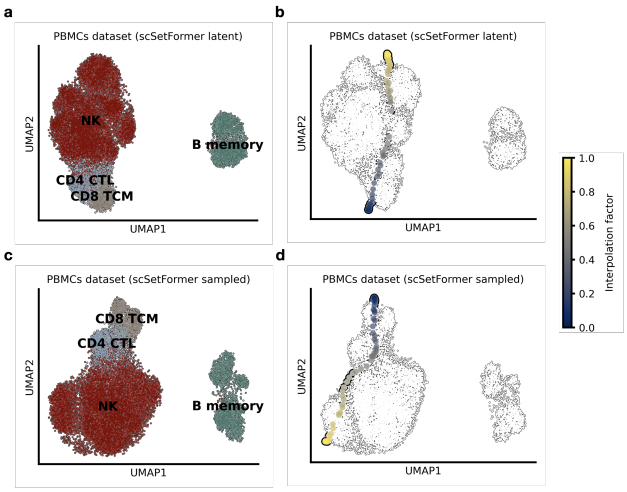

Figure 9. **Insights into the latent space of scSetFormer for the PBMCs dataset.** (a) Projection of latent vectors $z \sim q(z|x)$ coloured by the cell type of the input cell. (b) Latent space interpolation from a latent vector of NK to CD8 T cells. (c) Sampled cells coloured by their cell type, as a reference for the generated cells using the interpolated latent vectors (d). Interpolated latent vectors and the corresponding generated cells are coloured in (b) and (d). All other latent vectors and cells are coloured white.

scSetFormer shows a smooth and an interpolatable latent space, where both the interpolated latent vectors (Fig. 9b) and the resulting gene expression profiles (Fig. 9d) follow a definite path in the UMAP from the NK to the CD8 TCM cells, with small steps caused by the interpolation factor α.

4.9. Biology preservation

The main goal of scSetFormer is to generate cells with realistic gene expression profiles. To analyse whether the generated cells retain the underlying biology of their conditioning cells, we perform three of the most common scRNA-seq downstream analyses: gene mapping, marker gene detection and gene set enrichment analysis.

Gene mapping The first analysis involves assessing whether the genes known to be highly expressed in cells of a particular cell type are conserved in the generated cells. This is typically done to map a cell type into clusters based on the expression levels of such pre-defined genes. A literature search suggests that the gene MS4A1 is highly expressed in B cells (Zuccolo et al., 2013), CD8B in $CD8^+$ T cells (Yang & Parkhouse, 1997), KLRC1 in NK T cells (Russick et al., 2020) and IL7R in both $CD4^+$ and $CD8^+$ T cells (Zaunders et al., 2014). We are using these genes to see if they can resemble the cells in which they should be highly expressed.

Our results show that the cell types can be mapped to the generated cells by scSetFormer as the expression levels of the examined genes are highly expressed in the expected cell clusters and are highly similar to the real expression levels (Fig. 10, A11f), indicating maintenance of important biological information.

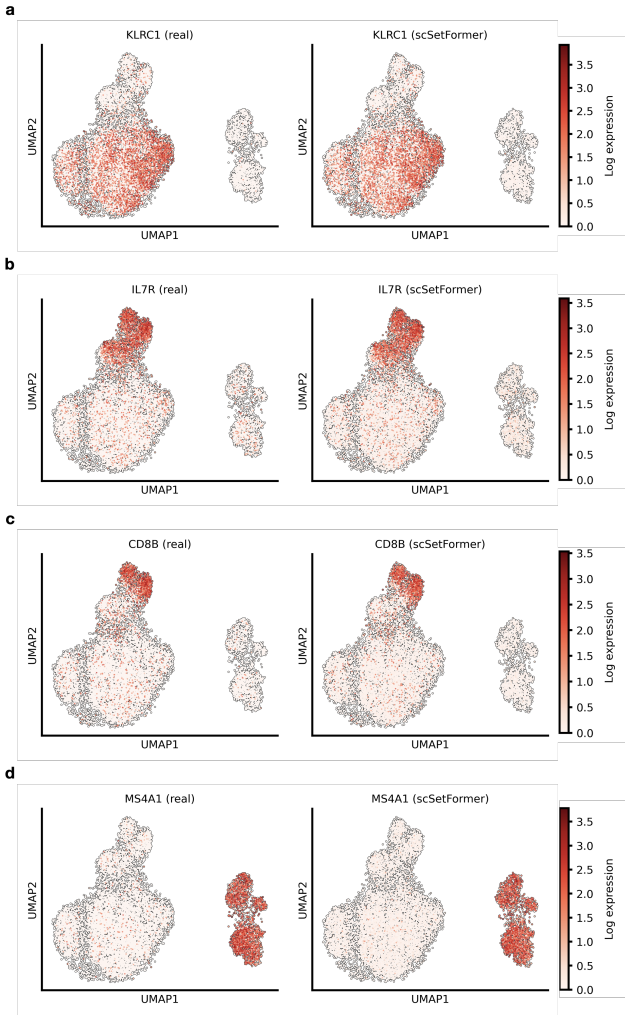

Figure 10. **Gene mapping for the PBMCs dataset.** Real and sampled cells are coloured according to the expression values of genes known to be highly expressed in the four corresponding cell clusters, i.e. KLRC1 for NK cells (a), IL7R for $CD4^+$ T cells (b), CD8B for $CD8^+$ T cells (c) and MS4A1 for B memory cells (d).

Marker gene detection The second analysis usually performed in scRNA-seq data analysis is to define a set of genes that are highly expressed in predefined cell clusters, also called *marker genes*. Here, we use scanpy to separately detect the top ten marker genes of each cell cluster t in the real cells ($\mathcal{M}_x^t$) as well as in the sampled cells ($\mathcal{M}_{\hat{x}}^t$). We then compute the Jaccard index for corresponding cluster pairs of type t as a similarity measure for the marker genes in both real and sampled cells. The Jaccard index J ranges from [0, 1], where 0 indicates no common marker genes and 1 indicates that both sets have exactly the same marker genes:

$$J(t) = \frac{|\mathcal{M}_{\hat{x}}^t \cap \mathcal{M}_x^t|}{|\mathcal{M}_{\hat{x}}^t \cup \mathcal{M}_x^t|}.$$

Genes in $\mathcal{M}_{\hat{x}}^t$, e.g. CD3D, show slightly higher values in cells where they should be close to zero, as shown for the $\mathcal{M}_{\hat{x}}^t$ in the real cells (Fig. 11a). While this over-expression is observed for all detected marker genes in $\mathcal{M}_{\hat{x}}^t$, their expression levels deviate only slightly from their expected expression levels. Marker genes in $\mathcal{M}_x^t$ with high expression levels, e.g. CD74, are also highly expressed in the generated cells (Fig. 11a).

The sampled B memory, NK and CD8 TCM cells retain most of the real marker genes in both conditional modes (Fig. 11b, A11e). The sampled CD4 CTL cells miss about 50% of the real marker genes in both conditional modes. This loss could be caused by the excess expression levels in the mono-conditional mode that have influenced the gene expression profiles, leading to a deviation from the real expression profiles. Additionally, this could be caused by the smaller number of cells of this type that scSetFormer encounters during training and may have led the model to focus more on learning the gene expression profiles of cell types with larger sample sizes.

Gene set enrichment analysis Finally, we examine the set of enriched pathways for each cell type in the real cells compared to the generated cells. During their development, cells activate different pathways depending on their type or current state of development. A pathway is the set of genes that are either downregulated, i.e. these genes are either switched off or transcribed at low levels in the cell and therefore the expression levels of these genes are low, or upregulated, i.e. the expression levels of these genes are high compared to other genes. It is therefore very important that the resulting cells have the same downregulated or upregulated gene pathways as their conditioning cells. Here we investigate which gene pathways are upregulated in the cell clusters in the real cells. To do this, we first define the set of upregulated genes in each cell type cluster in the real and generated cells separately as the marker genes identified in the marker gene detection part with a log fold change > 0.25 and with an adjusted p-value < 0.05, which reflects the significance of the differential expression across cell clusters. Finally, we use the Python package gseapy (Fang et al., 2023) to determine the top 5 gene ontology (GO) biological pathways in which the upregulated genes from each cluster are enriched, i.e. significantly enriched in

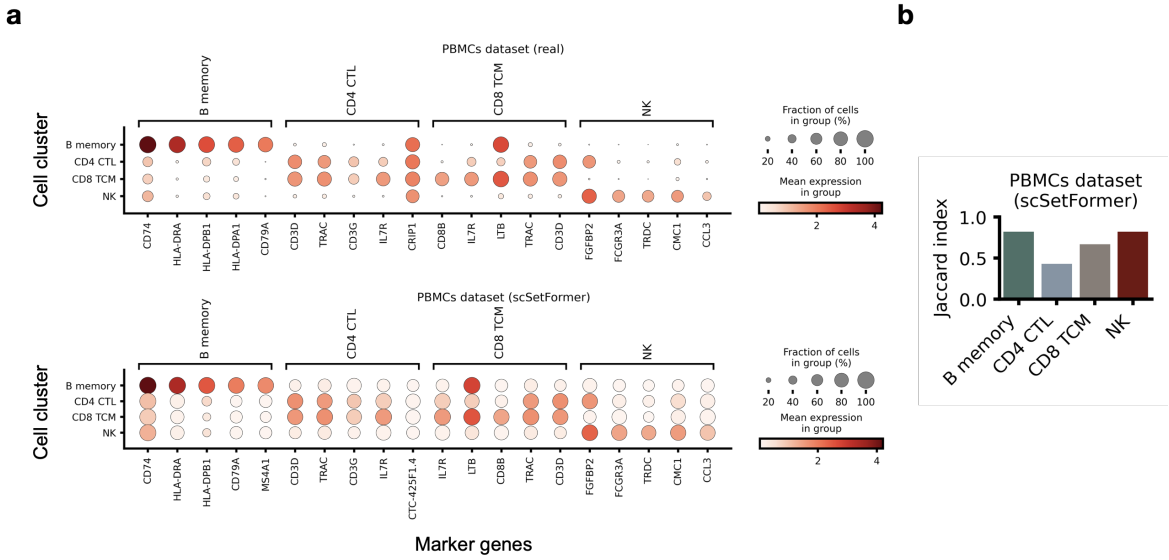

Figure 11. **Marker genes for the PBMCs dataset.** (a) Marker genes determined separately for each cell cluster in the real (top) and sampled (bottom) cells. Grouped marker genes on the x-axis correspond to the detected marker genes for the corresponding cell cluster. The size of the dots of a gene corresponds to the fraction of cells in each cell cluster, shown on the y-axis, expressing that gene, while the colour indicates the mean expression value in all cells belonging to that cluster. (b) Jaccard index for the overlapping marker genes in the real and sampled cells for each cell cluster.

Table 4. **Gene set enrichment analysis for the PBMCs dataset.** Top 5 gene pathways that are significantly enriched in the real B memory cells based on the top ten significantly upregulated differential genes. The first number is the number of upregulated genes in the real B memory cells and the second number is the number of genes in the respective pathway.

Gene pathway	Gene overlap
Cytokine-mediated signalling pathway	8/621
Antigen receptor-mediated signalling pathway	8/185
Antigen processing and presentation of exogenous peptide antigen via MHC class II	7/98
Antigen processing and presentation of exogenous peptide antigen	7/103
Antigen processing and presentation of peptide antigen via MHC class II	7/100

Table 5. **Gene set enrichment analysis for the PBMCs dataset.** Top 5 gene pathways that are significantly enriched in the B memory cells generated by scSetFormer based on the top ten significantly upregulated differential genes. The first number is the number of upregulated genes in the generated B memory cells and the second number is the number of genes in the respective pathway.

Gene pathway	Gene overlap
Cytokine-mediated signalling pathway	9/621
Antigen receptor-mediated signalling pathway	8/185
Antigen processing and presentation of exogenous peptide antigen via MHC class II	7/98
Antigen processing and presentation of exogenous peptide antigen	7/103
Antigen processing and presentation of peptide antigen via MHC class II	7/100

certain pathways.

The cells generated by scSetFormer show a preservation of the enriched pathways in the real cells, e.g. antigen presenting pathways in B memory cells (Tab. 4, 5), cellular response to interferon-gamma in NK cells (Tab. A1, A2), positive selection of T cells in the thymus in both CD4 CTL and CD8 TCM cells (Tab. A5, A6).

5. Discussion

Here, we introduced scSetFormer, a Set Transformer-based VAE for the conditional generation of scRNA-seq data. scSetFormer generates new cells conditioned not only on one input cell, like all current scRNA-seq data generative models, but also on multiple input cells. We characterise these possible conditioning modes by the mono-conditional and multi-conditional mode. The set attention mechanisms built

into the scSetFormer framework allow gene and cell order independent generation, removing any position bias caused by vanilla Transformers. Additionally, the attention mechanisms allow scSetFormer to exploit information aggregation across genes and cells, making the generation process more informative, and applicable to the generation of single or multiple cells that rely on gene expression profiles of multiple cells. scSetFormer in its mono-conditional mode shows similar performance to and sometimes outperforms scVI and scDiffusion. Additionally, data generated by scSetFormer maintain biological information. Furthermore, scSetFormer in its multi-conditional mode can still be improved to generate cells with more realistic gene expression profiles. The different conditional modes, both the mono- and multi-conditional mode, make scSetFormer more flexible and promote it as a foundation for the various *in silico* scRNA-seq data generation tasks.

Related work The *in silico* generation of scRNA-seq data has been explored using numerous approaches. Splatter (Zappia et al., 2017) is a simulation-based method using a hierarchical gamma-Poisson model. It simulates the mean expression of each gene from a gamma distribution and cell counts from a Poisson distribution. Synthesis using geometrically aligned random-walks (SUGAR, Lindenbaum et al. 2018) constructs a diffuse geometry around each data point is constructed using a Gaussian kernel. New data points are sampled using a sparsity-based measure. Single-cell variational inference (scVI, Lopez et al. 2018) uses a variational inference with six dense layers to model the gene expression distribution. Single-cell generative adversarial network (scGAN, Marouf et al. 2020) employs a generative adversarial network framework, where a generator and a discriminator model are trained to generate new cells that should be indistinguishable from real cells. scGAN allows cell type conditional generation of new cells. The latest generative model is scDiffusion (Luo et al., 2024), which explores the generation of scRNA-seq data based on a diffusion model. With its trained cell type classifier, scDiffusion, similar to scGAN, also allows the generation of new cells given a specific cell type label.

Future work scSetFormer is very flexible and could, therefore, be extended and applied to many *in silico* scRNA-seq data generation tasks in numerous forms. scSetFormer in its current setup and with the vanilla set attention modules, which scale quadratically with the number of input tokens, requires a long training time. Therefore, a more efficient attention can be applied instead to reduce the training time complexity. Due to the quadratic time complexity of the attention mechanism, here we train and generate new cells based only on the top 100 highly variable genes. scSetFormer could be optimised for all or more highly variable genes. scSetFormer generates excessive gene expression levels, which may be mitigated by incorporating an L1 regularisation applied to the generated cells. The generation of new cells based on multiple input cells sometimes results in gene expression profiles that are less similar to the input cells. More extensive hyperparameter search can be applied to enhance the performance of scSetFormer in this mode. Furthermore, the set of input cells in the multi-conditional mode is restricted to belong to the same cell type. While this is desirable in some applications, scSetFormer can be extended to generate new cells based on a heterogeneous input set of cells. Finally, in all our experiments, we trained and evaluated scSetFormer on static scRNA-seq data only. scSetFormer can, however, be applied to dynamic scRNA-seq data, with many possible applications in the multi-conditional mode.

References

Andoni, A., Indyk, P., and Krauthgamer, R. Earth mover distance over high-dimensional spaces. In *SODA*, volume 8, pp. 343–352, 2008.

Barrow, H., Tenenbaum, J., Hanson, A., and Riseman, E. Recovering intrinsic scene characteristics. *Comput. vis. syst*, 2(3-26):2, 1978.

Berthelot, D., Raffel, C., Roy, A., and Goodfellow, I. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*, 2018.

Bischoff, S., Darcher, A., Deistler, M., Gao, R., Gerken, F., Gloeckler, M., Haxel, L., Kapoor, J., Lappalainen, J. K., Macke, J. H., et al. A practical guide to statistical distances for evaluating generative models in science. *arXiv preprint arXiv:2403.12636*, 2024.

Bonneel, N., Rabin, J., Peyré, G., and Pfister, H. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51:22–45, 2015.

Bozkurt, A., Esmaeili, B., Tristan, J.-B., Brooks, D. H., Dy, J. G., and van de Meent, J.-W. Rate-regularization and generalization in vaes. *arXiv preprint arXiv:1911.04594*, 2019.

Breiman, L. Random forests. *Machine learning*, 45:5–32, 2001.

Chen, G., Ning, B., and Shi, T. Single-cell rna-seq technologies and related computational data analysis. *Frontiers in genetics*, 10:317, 2019.

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., and Shanahan, M. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Efremova, M., Vento-Tormo, R., Park, J.-E., Teichmann, S. A., and James, K. R. Immunology in the era of single-cell technologies. *Annual Review of Immunology*, 38: 727–757, 2020.

Fang, Z., Liu, X., and Peltz, G. Gseapy: a comprehensive package for performing gene set enrichment analysis in python. *Bioinformatics*, 39(1):btac757, 2023.

Gayoso, A., Lopez, R., Xing, G., Boyeau, P., Valiollah Pour Amiri, V., Hong, J., Wu, K., Jayasuriya, M., Mehlman, E., Langevin, M., et al. A python library for probabilistic analysis of single-cell omics data. *Nature biotechnology*, 40(2):163–166, 2022.

Gokcesu, K. and Gokcesu, H. Generalized huber loss for robust learning and its efficient minimization for a robust statistics. *arXiv preprint arXiv:2108.12627*, 2021.

Gong, J., Hao, M., Cheng, X., Zeng, X., Liu, C., Ma, J., Zhang, X., Wang, T., and Song, L. xtrimogene: An efficient and scalable representation learner for single-cell rna-seq data. *Advances in Neural Information Processing Systems*, 36, 2024.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Hahaut, V., Pavlinic, D., Carbone, W., Schuierer, S., Balmer, P., Quinodoz, M., Renner, M., Roma, G., Cowan, C. S., and Picelli, S. Fast and highly sensitive full-length single-cell rna sequencing using flash-seq. *Nature biotechnology*, 40(10):1447–1451, 2022.

Harshvardhan, G., Gourisaria, M. K., Pandey, M., and Rautaray, S. S. A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38:100285, 2020.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jovic, D., Liang, X., Zeng, H., Lin, L., Xu, F., and Luo, Y. Single-cell rna sequencing technologies and applications: A brief overview. *Clinical and Translational Medicine*, 12(3):e694, 2022.

Khan, M. Z., Jabeen, S., Khan, M. U. G., Saba, T., Rehmat, A., Rehman, A., and Tariq, U. A realistic image generation of face from text description using the fully trained generative adversarial networks. *IEEE Access*, 9:1250–1260, 2020.

Kim, D. and Suh, B. Enhancing vaes for collaborative filtering: flexible priors & gating mechanisms. In *Proceedings of the 13th ACM conference on recommender systems*, pp. 403–407, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kolouri, S., Pope, P. E., Martin, C. E., and Rohde, G. K. Sliced-wasserstein autoencoder: An embarrassingly simple generative model. *arXiv preprint arXiv:1804.01947*, 2018.

Kramer, O. and Kramer, O. Scikit-learn. *Machine learning for evolution strategies*, pp. 45–53, 2016.

Kulkarni, A., Anderson, A. G., Merullo, D. P., and Konopka, G. Beyond bulk: a review of single cell transcriptomics methodologies and applications. *Current opinion in biotechnology*, 58:129–136, 2019.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Lee, M. Recent advances in generative adversarial networks for gene expression data: a comprehensive review. *Mathematics*, 11(14):3055, 2023.

Li, W. Whole-embryo scrna-seq of mouse development. *Nature Genetics*, pp. 1–1, 2023.

Lindenbaum, O., Stanley, J., Wolf, G., and Krishnaswamy, S. Geometry based data generation. *Advances in Neural Information Processing Systems*, 31, 2018.

Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., and Yosef, N. Deep generative modeling for single-cell transcriptomics. *Nature methods*, 15(12):1053–1058, 2018.

Lopez-Paz, D. and Oquab, M. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

Luo, E., Hao, M., Wei, L., and Zhang, X. scdiffusion: conditional generation of high-quality single-cell data using diffusion model. *arXiv preprint arXiv:2401.03968*, 2024.

Marouf, M., Machart, P., Bansal, V., Kilian, C., Magruder, D. S., Krebs, C. F., and Bonn, S. Realistic in silico generation and augmentation of single-cell rna-seq data using generative adversarial networks. *Nature communications*, 11(1):166, 2020.

Martinelli, D. D. Generative machine learning for de novo drug discovery: A systematic review. *Computers in Biology and Medicine*, 145:105403, 2022.

McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Pinheiro Cinelli, L., Araújo Marins, M., Barros da Silva, E. A., and Lima Netto, S. Variational autoencoder. In *Variational Methods for Machine Learning with Applications to Deep Networks*, pp. 111–149. Springer, 2021.

Russick, J., Joubert, P.-E., Gillard-Bocquet, M., Torset, C., Meylan, M., Petitprez, F., Dragon-Durey, M.-A., Marmier, S., Varthaman, A., Josseaume, N., et al. Natural killer cells in the human lung tumor microenvironment display immune inhibitory functions. *Journal for Immunotherapy of Cancer*, 8(2), 2020.

Schölkopf, B. The kernel trick for distances. *Advances in neural information processing systems*, 13, 2000.

Shaham, U., Stanton, K. P., Zhao, J., Li, H., Raddassi, K., Montgomery, R., and Kluger, Y. Removal of batch effects using distribution-matching residual networks. *Bioinformatics*, 33(16):2539–2546, 2017.

Shahapure, K. R. and Nicholas, C. Cluster quality analysis using silhouette score. In *2020 IEEE 7th international conference on data science and advanced analytics (DSAA)*, pp. 747–748. IEEE, 2020.

van den Burg, G. and Williams, C. On memorization in probabilistic deep generative models. *Advances in Neural Information Processing Systems*, 34:27916–27928, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wen, L., Li, G., Huang, T., Geng, W., Pei, H., Yang, J., Zhu, M., Zhang, P., Hou, R., Tian, G., et al. Single-cell technologies: From research to application. *The Innovation*, 3(6), 2022.

Wolf, F. A., Angerer, P., and Theis, F. J. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19:1–5, 2018.

Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., and Tang, Y. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136, 2023.

Xu, H., Chen, W., Lai, J., Li, Z., Zhao, Y., and Pei, D. On the necessity and effectiveness of learning the prior of variational auto-encoder. *arXiv preprint arXiv:1905.13452*, 2019.

Yang, H. and Parkhouse, R. Differential expression of cd8 epitopes amongst porcine cd8-positive functional lymphocyte subsets. *Immunology*, 92(1):45–52, 1997.

Yazar, S., Alquicira-Hernandez, J., Wing, K., Senabouth, A., Gordon, M. G., Andersen, S., Lu, Q., Rowson, A., Taylor, T. R., Clarke, L., et al. Single-cell eqtl mapping identifies cell type–specific genetic control of autoimmune disease. *Science*, 376(6589):eabf3041, 2022.

Zappia, L., Phipson, B., and Oshlack, A. Splatter: simulation of single-cell rna sequencing data. *Genome biology*, 18(1):174, 2017.

Zaunders, J. J., Lévy, Y., and Seddiki, N. Exploiting differential expression of the il-7 receptor on memory t cells to modulate immune responses. *Cytokine & growth factor reviews*, 25(4):391–401, 2014.

Zheng, G. X., Terry, J. M., Belgrader, P., Ryvkin, P., Bent, Z. W., Wilson, R., Ziraldo, S. B., Wheeler, T. D., McDermott, G. P., Zhu, J., et al. Massively parallel digital transcriptional profiling of single cells. *Nature communications*, 8(1):14049, 2017.

Zuccolo, J., Deng, L., Unruh, T. L., Sanyal, R., Bau, J. A., Storek, J., Demetrick, D. J., Luider, J. M., Auer-Grzesiak, I. A., Mansoor, A., et al. Expression of ms4a and tmem176 genes in human b lymphocytes. *Frontiers in immunology*, 4:195, 2013.

Appendix

A1. Metrics evaluation

In one experiment, we encountered contradictory conclusions about the quality of the generated data when benchmarked using the MMD distance with a Gaussian kernel and in addition to the C2ST. In this experiment, we trained a one-component Gaussian mixture model (GMM) to learn the underlying distribution of the PBMCs dataset (Yazar et al., 2022). While the MMD value between the real and generated data using this model was 1.90e-05, which is very close to the baseline MMD value of 1.7e-06, the C2ST resulted in an accuracy of 1.0. Additionally, projection of the real and generated data in a UMAP showed no coverage of the B memory cell cluster by the generated cells (Fig. A1b), which is explained by the choice of a one-component GMM with . This highlights a potential failure of the MMD distance to reflect the true difference between two distributions.

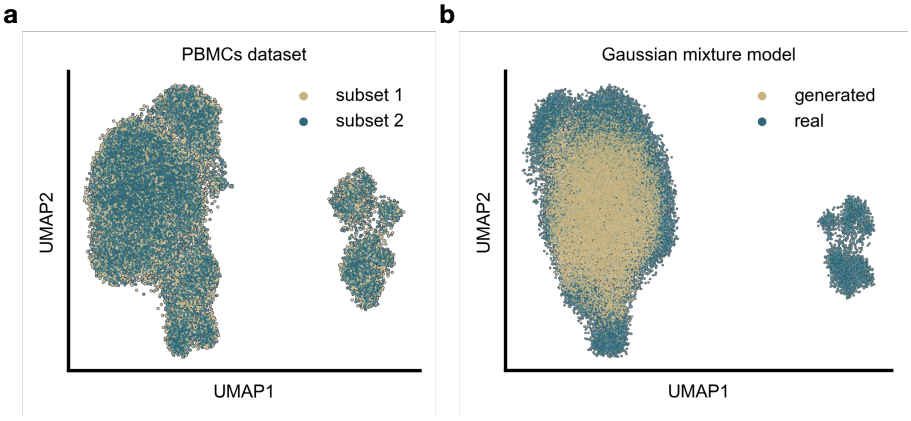

Figure A1. **Real and sampled cells projected by UMAP for the PBMCs dataset.** (a) Real cells divided into two subsets 1 and 2. (b) Real and sampled cells by a Gaussian mixture model.

To better understand the underlying mechanism of the MMD, we evaluate the MMD on two toy datasets (Fig. A2). The first one should represent the real cells (dark blue points) and the second one should represent the generated cells (dark yellow points). All data points are drawn from a 2D isotropic Gaussian. To simulate differences in the underlying distributions, points from both toy datasets are pulled in four directions and translated by a certain factor, which we call here *imbalance*. The impact of the imbalance factor on the translated points can be seen from the shape of the data and the increasing x- and y-axis ranges with an increasing imbalance (Fig. A2).

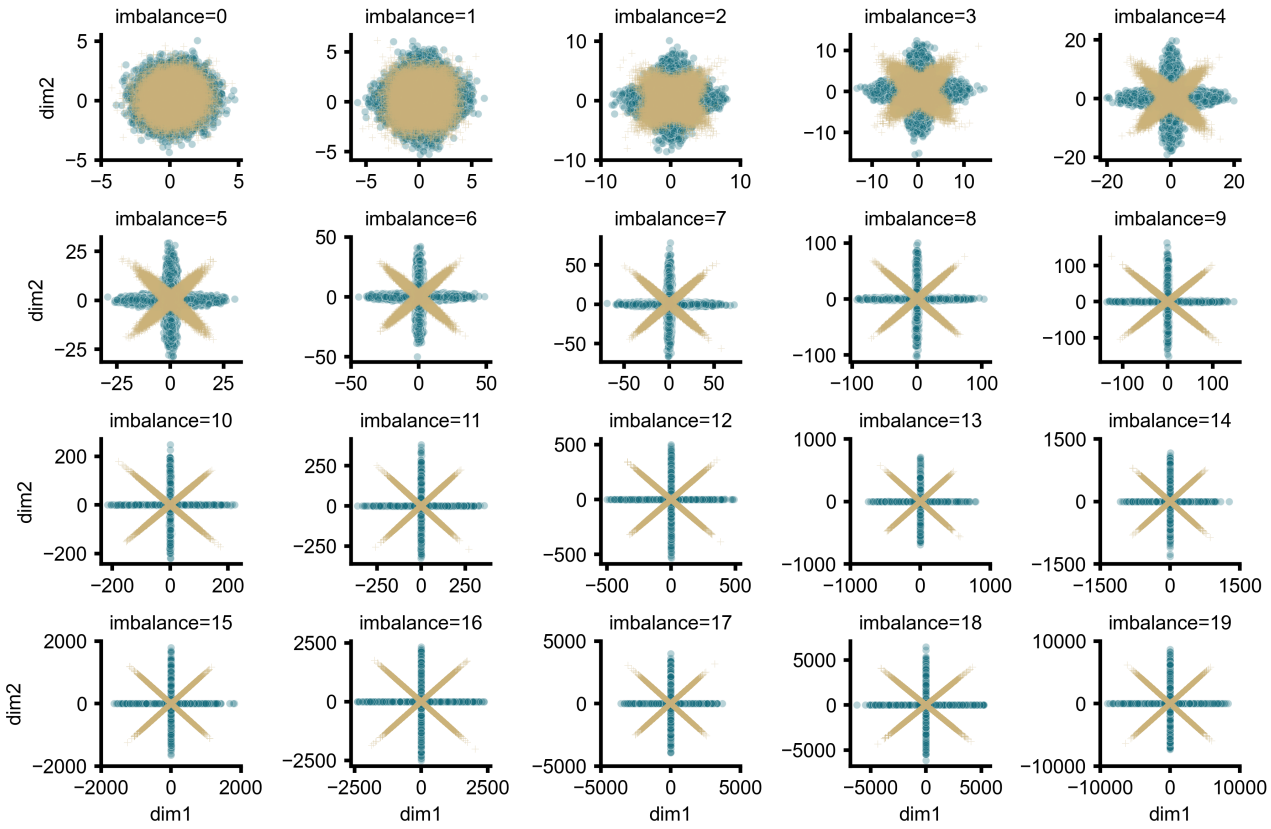

Figure A2. **2D toy dataset with two clusters.** Points in cluster 0 (orange) are drawn from a 2D Gaussian with means, both set to 5, and standard deviation set to 1. Points in cluster 1 (brown) are drawn from a 2D Gaussian with means, both set to 10, and standard deviation set to 1.

The imbalance should, therefore, give a direct measure of how dissimilar the two toy datasets are. Furthermore, we apply the MMD to the two toy datasets over 20 increasing imbalances. As we use a Gaussian kernel, we apply different γ, a hyperparameter that controls the bandwidth of the Gaussian kernel, to mimic the influence of the bandwidth on the MMD performance. A high γ value corresponds to a narrow bandwidth. In this case, the Gaussian kernel only considers the very close neighbouring data points of the point of interest. A low γ value indicates a wide bandwidth, which allows the Gaussian kernel to consider data points that are further away from the point of interest. We further apply a similar procedure to the PBMCs dataset. Here, we randomly select a subset of the cells and shift all their gene expression values by an imbalance factor in the range 0 to 1, as the data is normalised and log-transformed. For both the toy and PBMCs datasets, we generally expect an increasing MMD distance with an increasing imbalance.

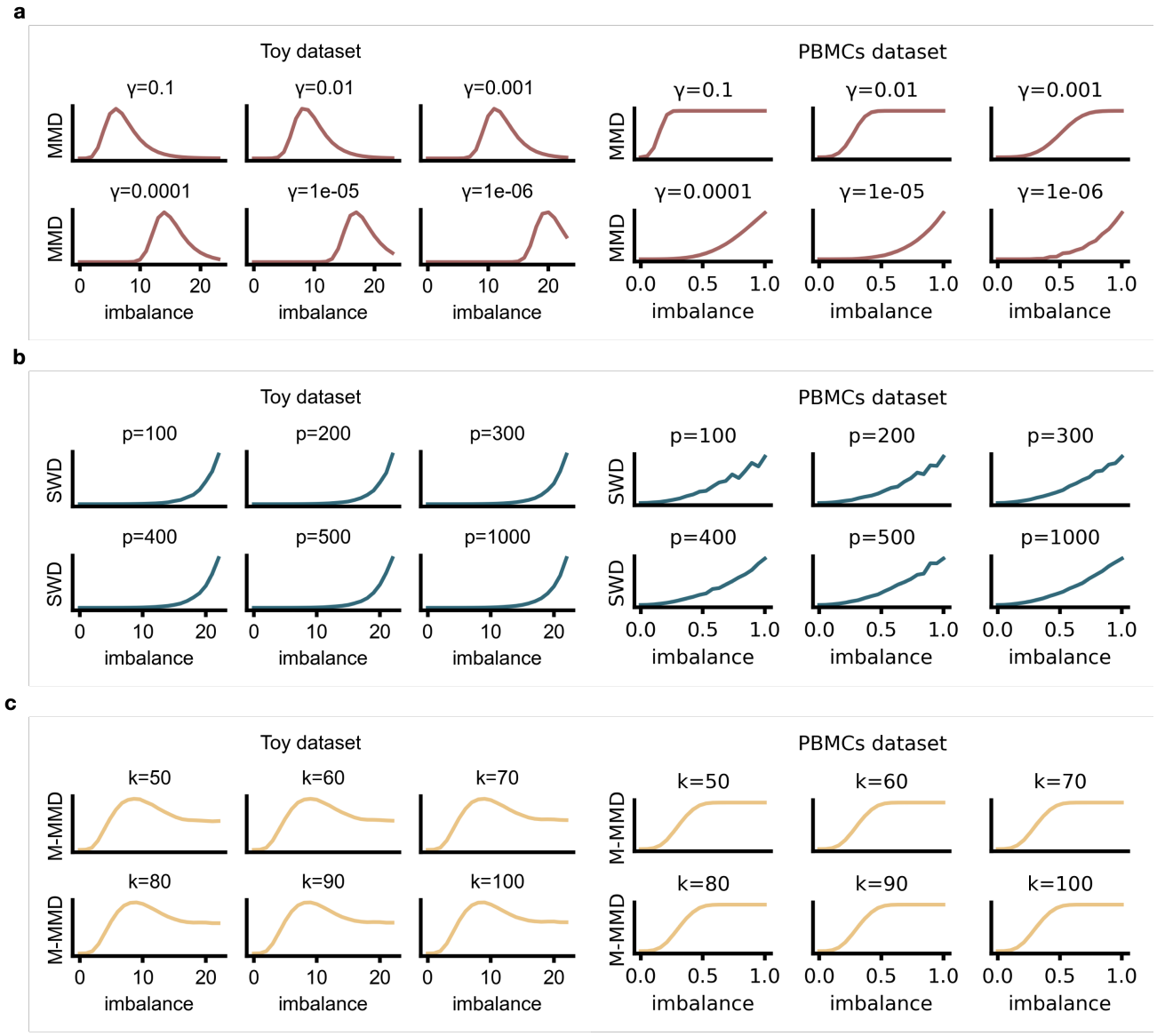

Figure A3. **Evaluation of MMD, SWD and M-MMD with different hyperparameters.** The metrics are applied to the 2D toy (left column) and the PBMCs dataset (right column). MMD (a), multi-kernel MMD (b) and SWD (c) are applied to the translated data over 20 increasing imbalance factors. The value of the corresponding hyperparameter is shown in the subtitles.

Our results on the toy dataset show that the choice of γ has a high impact on the resulting MMD distances (Fig. A3a, left). High γ values show small MMD distances for increasingly dissimilar data points. This effect can be seen by the drop of the MMD distances after a certain imbalance factor is reached. While this effect decreases with decreasing γ, the MMD still show the same behaviour.

This behaviour could be caused by how many points does the Gaussian kernel encounters simultaneously from both the real and generated datasets, which is influenced by its bandwidth. A Gaussian kernel with a narrow bandwidth, i.e. high γ, may only encounter points from one of the two datasets and focus on local features of the data distributions, resulting in smaller MMD distances for datasets that are largely dissimilar or have distributions with multiple and different modes, corresponding to the four point clusters that are spread in four different directions in the toy datasets, but share points originating from a small number of modes, corresponding to the points around the centre from both toy datasets.

Gaussian kernels with a wider bandwidth, i.e. smaller γ, can capture more points from both datasets at the same time, and thus a broader feature of the data distributions. Since decreasing the γ did not prevent the failure of MMD to reflect the imbalance between the datasets, an even smaller γ may help to prevent it altogether. We, therefore, assume that the choice of γ depends on the underlying distribution of the dataset of interest.

Applying Gaussian kernels with smaller γ shows an increasing MMD with increasing imbalance, which is the expected behaviour (Fig. A3a, right). The behaviour of the MMD on this dataset could be explained by the following: the cells generated by a generative model do not necessarily differ in all of their expression values from the real ones. Therefore, shifting all gene expression values of the selected cells by an imbalance factor, as done in this experiment, might not reflect the MMD behaviour on the cells generated by generative models. We, therefore, decide to assess the MMD distance using a different metric.

Since our scRNA-seq data are multimodal, we choose the sliced Wasserstein distance (SWD, Bonneel et al. 2015), which is used in GANs to mitigate the mode collapse problem, as it is designed to cover dissimilarities between all data points from both data distributions. We apply the SWD with different number of projections p to both the toy and PBMCs datasets. The SWD successfully shows an exponential and a linear increase with increasing imbalance on the toy and PBMCs datasets, respectively (Fig. A3b). In addition, we examine the multi-kernel MMD (M-MMD, Shaham et al. (2017)), where the MMD value is computed as the sum of three different Gaussian kernels with γ equal to $m, 2m, \frac{m}{2}$ and $2m$, where m is the median of the average distance between a cell and its k nearest neighbours. We apply the M-MMD with different values of k. The M-MMD shows a similar behaviour to the MMD on the toy dataset, where it decreases after reaching a certain imbalance, but then stabilises (Fig. A3c, left). On the PBMCs dataset, it does not decrease but remains constant after reaching an imbalance factor of 0.5 (Fig. A3c, right).

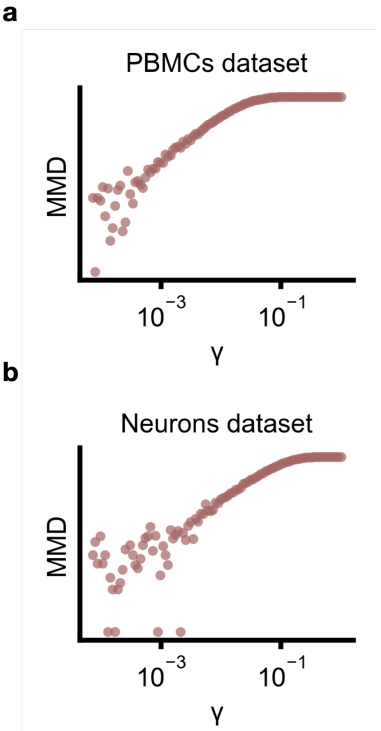

Figure A4. **Determining the optimal bandwidth of the Gaussian kernel.** MMD with a Gaussian kernel is applied to the PBMCs (top) and neurons (bottom) datasets with different values of γ, which controls the bandwidth of the Gaussian kernel.

As the M-MMD shows similar results to the MMD, we only include the SWD in the benchmarking metrics. Additionally, we choose 8.0e-05 and 2.8e-04 as values for γ for the PBMCs and neurons datasets, respectively. These values were chosen after applying the MMD with different γ values to the PBMCs (Fig. A4a) and neurons (Fig. A4b) datasets separately. Here we randomly divide the cells into two equally sized clusters, *subset 1* and *subset 2*. Since we expect the least difference between cells in these clusters, the γ value that results in the smallest MMD distance was chosen as the γ for the Gaussian kernel in all corresponding and subsequent experiments.

A2. scSetFormer with flexible priors

VAEs allow an unconditional generation of new data points x' by sampling a latent vector z from the prior distribution that is assumed over the latent vectors, typically an isotropic Gaussian $p(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, and passing it to the decoder $p_\phi(x|z)$. However, VAEs applied to more complex data, e.g. images, are known to produce unrealistic and blurred images (Pinheiro Cinelli et al., 2021). This problem is caused by so-called *posterior collapse*, where the latent prior distribution and the approximated latent posterior distribution are not matched by the KL divergence during training. Posterior collapse occurs because of the simplified assumption that the latent variables are drawn from an isotropic Gaussian, whereas the true latent posterior distribution is often a more complex distribution. As a result, latent vectors drawn from the prior distribution used to generate new data may have been assigned a low probability in the approximated posterior, and thus are unlikely to have been encountered by the decoder during training, resulting in the generation of unrealistic data points.

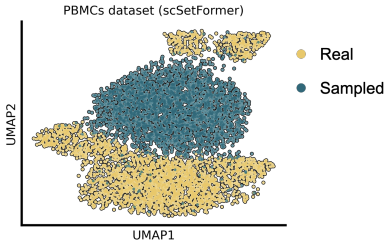

Figure A5. **Unconditional sampling using scSetFormer on the PBMCs dataset.** Projection of real and unconditionally sampled cells using latent vectors sampled from the latent prior $p(z)$, which is an isotropic Gaussian distribution.

As scSetFormer follows the general architecture of a VAE, we attempt to generate cells with new gene expression profiles unconditionally, i.e. without a conditioning input cell. Here, we encountered the problem of posterior collapse, where the generated cells did not show realistic gene expression profiles or overlapping spatial arrangement with the real cells (Fig. A5). Alternative priors are suggested for high quality generation of new data points (Xu et al., 2019). However, for most real-world data, the prior distribution of the underlying manifold, i.e. the latent space, is not known. So-called *flexible priors* are used to mitigate this problem. Flexible priors are not a fixed distribution, but are learned during training to match the approximated posterior, thus avoiding the posterior collapse problem. Many approaches have been proposed to learn the prior and the approximated posterior simultaneously (Kim & Suh, 2019). In this paper, we use two different approaches to learn the latent prior for more realistic unconditional generation of new cells: Gaussian mixture (Dilokthanakul et al., 2016) and a flow model based prior (Dinh et al., 2016). To better understand the behaviour of these approaches, we further apply them to a toy dataset consisting of two point clusters 0 and 1, each sampled from a 2D Gaussian distribution with means around 5 and 10, respectively, and a standard deviation of 1 for both (Fig. A6).

A2.1. Gaussian mixture-based variational autoencoders

In Gaussian mixture-based variational autoencoders (GMVAEs, Dilokthanakul et al. 2016), a Gaussian mixture distribution is assumed over the latent prior distribution, which is appropriate for data points drawn from a multimodal Gaussian distribution. If the class label of a data point is known or inferred via a classifier, the Gaussian mode from which the data point may have been sampled could be easily inferred, as each mode would correspond to a class label. GMVAEs are trained to optimise a more complex objective than vanilla VAEs. Let's assume the following:

- $p(w) = \mathcal{N}(0, I)$, a prior over latent vectors w.
- $p(y) = \text{Mult}(\pi)$, $\pi_i = \frac{1}{K}$, a prior over the label distribution of the K different labels in the dataset.
- $p_{\beta_z}(z|w, y) = \prod_{k=1}^{K} \mathcal{N}(\mu_{\beta_{y_k}}(w), \Sigma_{\beta_{y_k}}(w))^{y_k=1}$, a prior distribution over latent vectors z given latent vector w and label y.
- $q_{\phi_z}(z|x) = \mathcal{N}(\mu_{\phi_z}(x), \Sigma_{\phi_z}(x))$, the approximated posterior of latent vectors z given input data x and with learnable parameters ϕ.
- $q_{\phi_w}(w|x) = \mathcal{N}(\mu_{\phi_w}(x), \Sigma_{\phi_w}(x))$, the approximated posterior of latent vectors w given input data x and with learnable parameters ϕ.

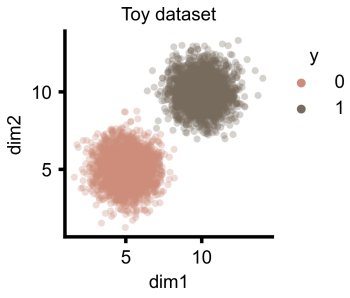

Figure A6. **2D toy dataset with two clusters.** y is the class label of the data points.

Then, the optimisation objective of the GMVAE is defined as follows:

$$\mathcal{L}(\theta, \phi, x) = \alpha \cdot \mathbb{E}_{z \sim q_{\phi_z}} [\log p_\theta(x|z)] \tag{1}$$

$$- \beta \cdot \mathbf{KL}[q_{\phi_w}(w|x) \| p(w)], \tag{2}$$

$$- \gamma \cdot \mathbb{E}_{w \sim q_{\phi_w}, y \sim q_{\beta_y}} [\mathbf{KL}[q_{\phi_z}(z|x) \| q_{\beta_z}(z|w, y)]] \tag{3}$$

$$- \delta \cdot \mathbb{E}_{z \sim q_{\phi_z}, w \sim q_{\phi_w}} [\mathbf{KL}[q_{\beta_y}(y|w, z) \| p(y)]] \tag{4}$$

where

- Term (1) corresponds to the reconstruction loss, it is the same as in vanilla VAEs.
- Term (2) corresponds to the KL divergence for the approximated posterior $q_{\phi_w}(w|x)$ over the latent variable w and its prior $p(w)$. It is optimised for generating new data points using one mode of the Gaussian mixture prior $p_{\beta_z}(z|w, y)$, determined by a given or sampled label y.
- Term (3) corresponds to the average KL divergence between the approximated posterior $q_{\phi_z}(z|x)$ and each component of the learned Gaussian mixture prior $q_{\beta_z}(z|w, y)$ weighted by the predicted probability y_k for each component k. This is equivalent to the KL divergence in vanilla VAEs, however, here K times KL divergences are computed and weighted by the corresponding label probability.
- Term (4) corresponds to the KL divergence between the approximated distribution over the predicted label vectors y q_{β_y} and their uniform prior distribution $p(y)$. This is used to optimise the prediction of a data point's label during training. It is most useful for data with unknown class labels.
- α, β, γ and δ are hyperparameter weighting factors.

In our experiments, we remove term (4) to make the optimisation objective less complex and due to the available labels of both the PBMCs and the toy data.

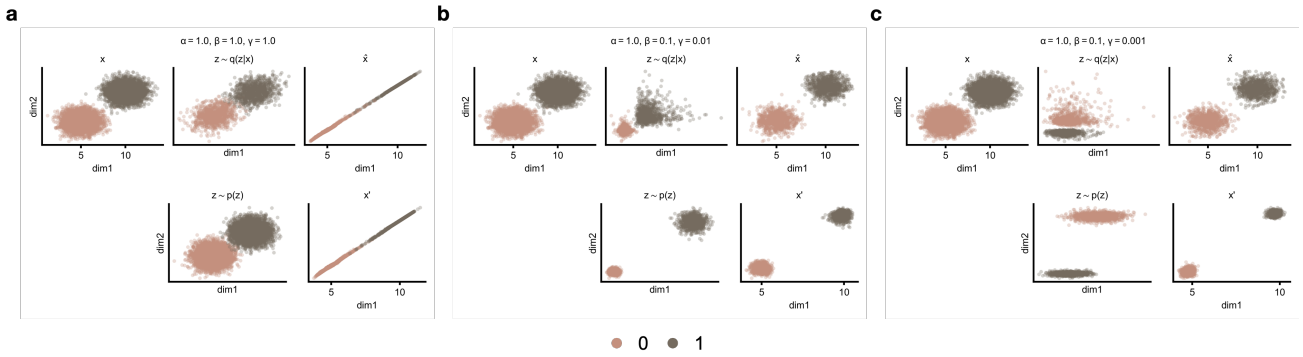

Figure A7. **Performance of scSetFormer with Gaussian mixture on the 2D toy dataset.** The performance based on different combinations of weighting factors α, β and γ is shown in (a), (b) and (c). The top row shows the generation of data points $\hat{x}$ conditioned on input points x with latent vectors z sampled from the approximated posterior distribution $q_\theta(z|x)$. The bottom row shows the generation path of new data points x' using latent vectors sampled from the prior distribution $p(z|y)$ for a given class label y.

Our results show a trade-off between mitigating the posterior collapse problem via the learnable prior $q_{\beta_z}(z|w, y)$ and minimising the reconstruction loss. Models with highly matched latent posteriors and priors suffer from variance loss in both the conditionally reconstructed $\hat{x}$ and the unconditionally generated data x'. This phenomenon can be observed in both the models trained on the PBMCs and the toy data sets. In the latter, the generated data points lack variance in one direction (Fig. A7a). In the former, the lack of variance was observed between cells belonging to one of the three T cell types. Here, although the model learns a bi-model Gaussian in the prior, one for the three T cell types and one for the B cell type, cells generated from the mode corresponding to the T cell cluster fail to preserve the variance between the three

different T cell types (Fig. A10a). While the unconditionally generated data points are better than those generated from an isotropic Gaussian in vanilla VAE, finding a balanced set of values for the α, β and γ weighting factors is a key to achieving even better quality data in both the conditional and unconditional modes as a result of the matched latent posterior and prior (Fig. A7b, c).

A2.2. Flow-based models

In flow-based models, a distribution is estimated in a data-driven manner. Here we use real-valued non-volume preserving (RealNVP, Dinh et al. 2016) to learn the latent prior. RealNVP allows the transformation of a simple distribution into a more complex one by learning a bijective mapping between the simple distribution, e.g. an isotropic Gaussian distribution, and the complex distribution, e.g. the approximated posterior $q_\theta(z|x)$.

Let $p_V(v)$ be an isotropic Gaussian base distribution corresponding to the simple distribution over the vector $v \in V$, let $z \sim q_\theta(z|x)$ be a latent vector from the complex distribution $q_\theta(z|x)$, and $f : V \rightarrow Z$ m inevitable and differentiable transformations, while a transformation f_i models the *change of variable*, i.e. the change from v to z, and thus f corresponds to a mapping function from the base distribution $p_V(v)$ to the approximated posterior $q_\theta(z|x)$. The prior $p(z)$ transformed by RealNVP can be written as follows:

$$\log p(z) = \log p_V(v) - \sum_{i}^{m} |\frac{\partial f_i(v_{i-1})}{\partial v_{i-1}}|, \tag{5}$$

where $|\frac{\partial f_i(v_{i-1})}{\partial v_{i-1}}|$ is the Jacobian determinant of the ith transformation used to sequentially model the change in volume between the base and complex distributions, where $f_i(v_{i-1}) = v_i$, $v_0 = v, v_m = z$. We can rewrite ELBO as follows:

$$\begin{aligned} \log p_\phi(x) &\geq \mathbb{E}_{q_\theta(z|x)} \left[\log \frac{p_\phi(x,z)}{q_\theta(z|x)} \right] \\ &= \mathbb{E}_{q_\theta(z|x)} \left[\log p_\phi(x|z) - \log q_\phi(z|x) + \log p(z) \right]. \end{aligned} \tag{6}$$

If we replace the fixed prior $p(z)$ in Eq. 6 with the transformed prior in Eq. 5, we obtain the following objective function,

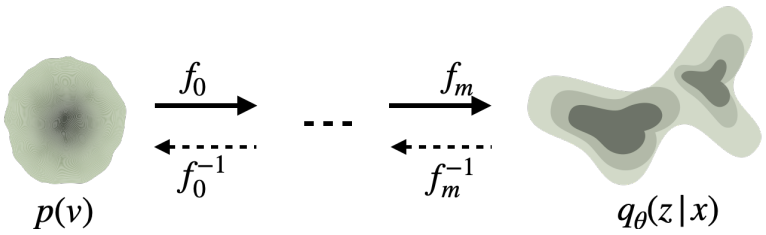

Figure A8. **Bijective transformations in RealNVP.** $p(v)$ represents the base distribution (an isotropic Gaussian) and $q_\theta(z|x)$ represents the approximated posterior distribution, which is of a more complex distribution.

which we use to train scSetFormer for unconditional generation of new data points:

$$\mathcal{L}(x;\phi,\theta) = \mathbb{E}_{q_\theta(z|x)} \left[\alpha \cdot \log p_\phi(x|z) - \beta \cdot (\log q_\phi(z|x) + \log p_V(v_0)) + \gamma \cdot \sum_{i}^{m} |\frac{\partial f_i^{-1}(v_i)}{\partial v_i}| \right],$$

where α, β and γ are hyperparameter weighting factors. The unconditional generation from through the prior $p(z)$ is then done using the transformed prior as follows:

- latent vector v is sampled from the base distribution $p_V(v)$,
- v is then passed to f to obtain the corresponding transformed latent vector z, and
- z is finally passed to the decoder to generate a new data point.

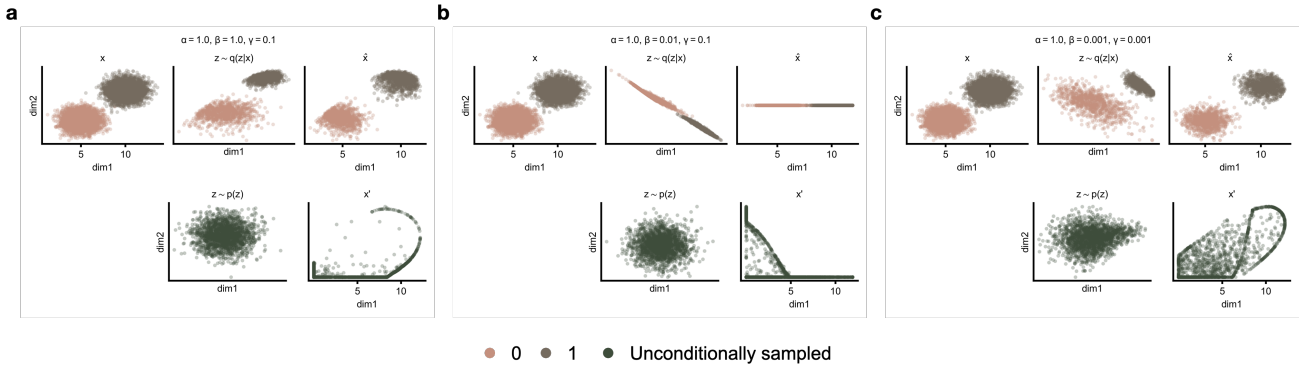

Figure A9. **Performance of scSetFormer with RealNVP on the 2D toy dataset.** The performance for different set weighting factors α, β and γ is shown in (a), (b) and (c). The top row shows the generation path of data points $\hat{x}$ conditioned on input points x with latent vectors z sampled from the approximated posterior distribution $q_\theta(z|x)$. The bottom row shows the generation path of new data points x' using latent vectors sampled from the prior distribution $p(z)$. The latent vectors z and the generated data points x' have no class labels assigned to them, as the sampling from the transformed prior using the RealNVP approach is unsupervised,unlike scSetFormer with Gaussian mixture prior.

Our results on both the PBMCs and toy datasets show that training a VAE with a prior learned via RealNVP leads to unstable learning. On the toy dataset, the model generates data points that are close to the real data (Fig. A9), but the prior fails to match the multi-model posterior distribution (Fig. A9). We observe the same behaviour on the PBMCs dataset (Fig. A10b). We believe that samples generated by the transformed latent prior using RealNVP could have a better quality with more training time and a larger number of transformation functions f_i. Similar to GMVAE, the weights applied to the different terms in the loss function have a huge impact on the quality of the generated data (Fig. A9). Finding a balanced set of weights is therefore crucial in this approach as well.

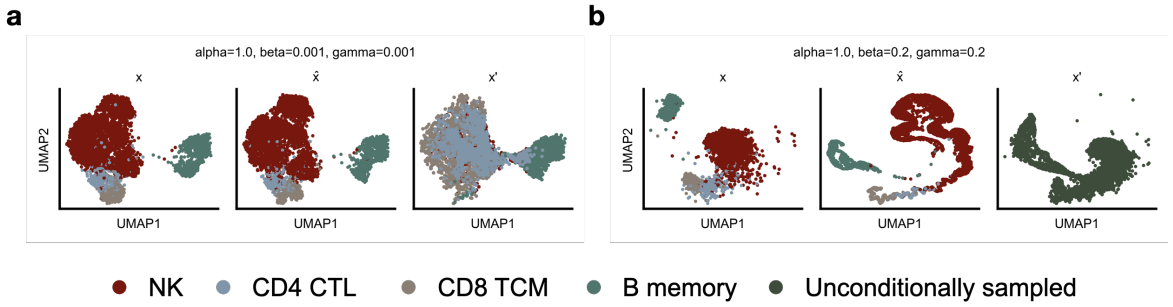

Figure A10. **Performance of the scSetFormer with flexible priors on the PBMCs dataset.** scSetFormer with Gaussian mixture (a) and RealNVP (b) as latent prior. Real cells (left), sampled cells (middle) with the approximated posterior distribution $q_\theta(z|x)$ and sampled cells with the prior distribution conditioned on an input label (right). The weighting factors for the loss terms are at the top. The sampled cells with latent vectors sampled from the RealNVP latent prior distribution have no cell type assigned to them as the latent vectors were sampled based on no conditions.

A3. Further results

A3.1. scSetFormer in multi-conditional mode

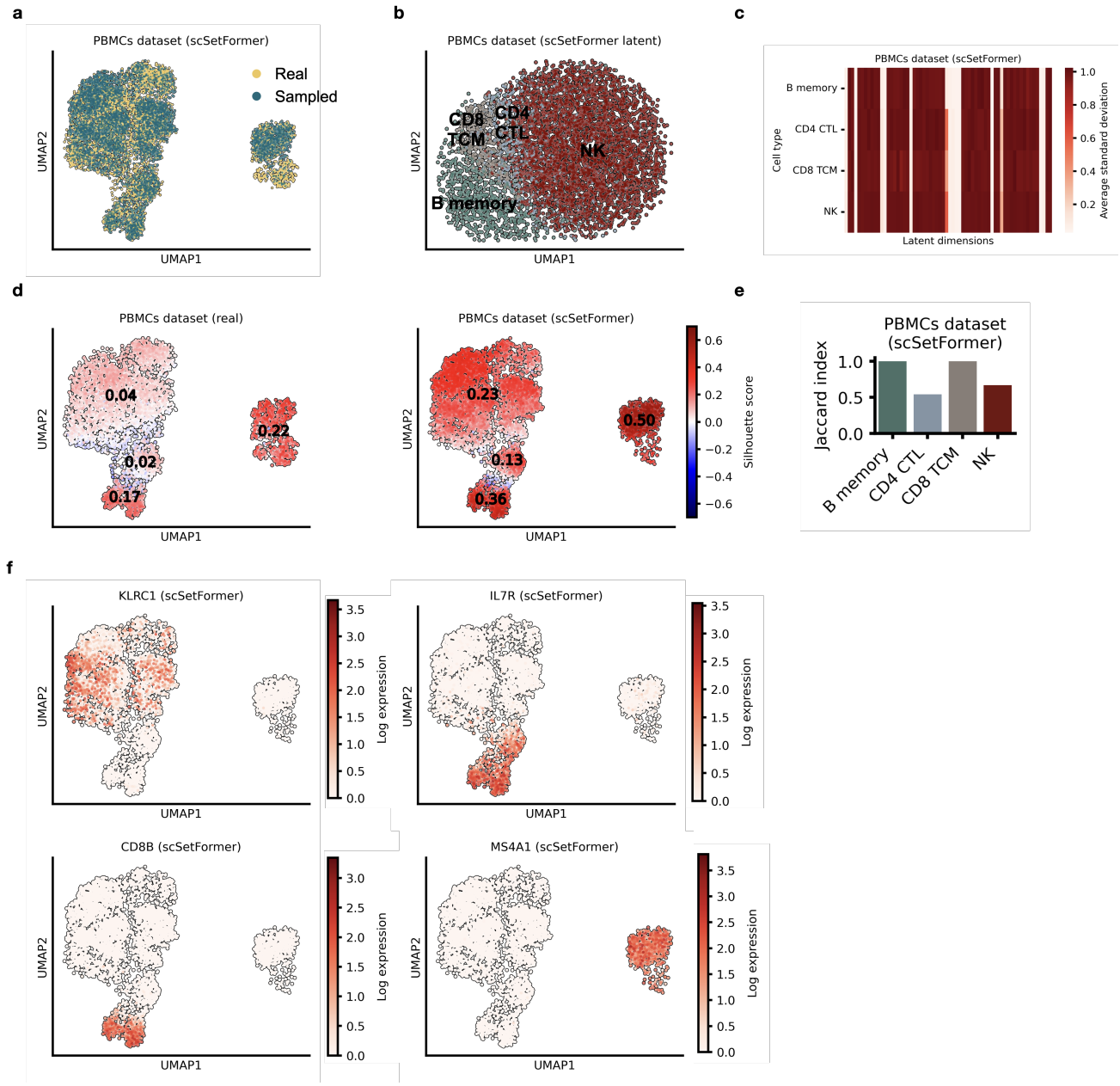

Figure A11. **Experimental results of scSetFormer in its multi-conditional, mono-generation mode.** Projection of real and sampled cells (a) and latent vectors coloured by the cell type of the input set of cells (b). (c) Heatmap showing the standard deviation for each latent dimension averaged over latent vector clusters determined by the cell type of the corresponding input set of cells. (d) UMAP with cells coloured according to their Silhouette score. The average of the Silhouette score for each cell cluster is shown on the corresponding cell cluster. The cells sampled by scSetFormer show higher Silhouette scores due to the larger number of real input cells compared to the sampled cells. (e) Jaccard index for the overlapping marker genes in the real and sampled cells. (f) Gene mapping of four marker genes.

A3.2. scSetFormer evaluation on neurons dataset

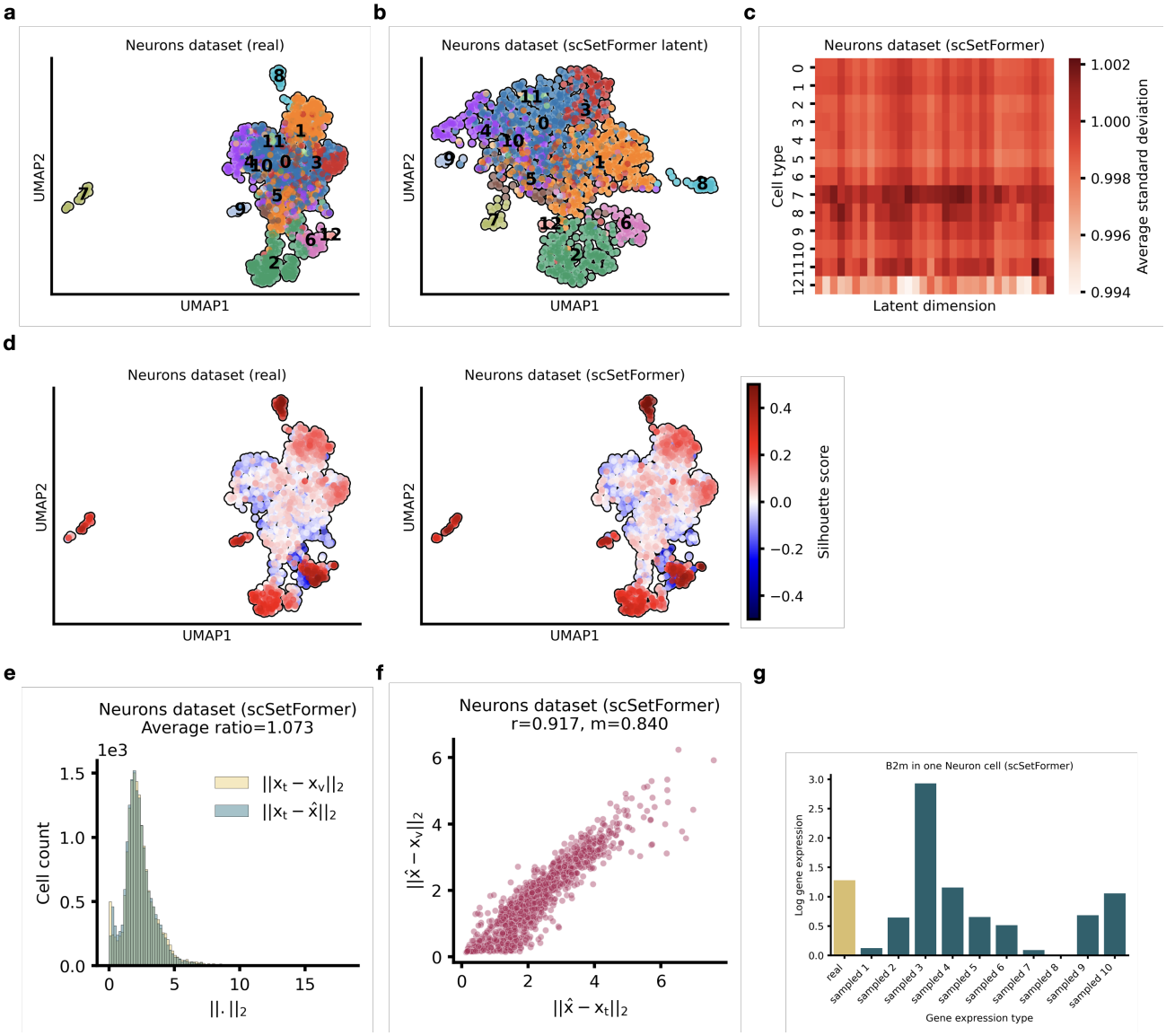

Figure A12. **Experimental results of scSetFormer in its mono-conditional mode on the neurons dataset.** Projection of real cells (a) and latent vectors coloured by the cell cluster of the input set of cells (b). (c) Heatmap showing the standard deviation for each latent dimension averaged over the latent vector clusters determined by the cell type of the corresponding input set of cells. (d) Cells coloured according to the Silhouette score. (e) Distribution of the Euclidean distance between the real cells in the training dataset x_t and the real cells in the validation dataset x_v (yellow) and the Euclidean distances between the training dataset and the generated cells $\hat{x}$ (blue). (f) Euclidean distances of the generated cells to the training dataset versus Euclidean distances between of the generated cells to the validation dataset. r corresponds to the Pearson correlation coefficient and m corresponds to the slope of a linear fit. (g) Real and sampled gene expression values for gene B2m in a cell.

A3.3. Memorisation checks

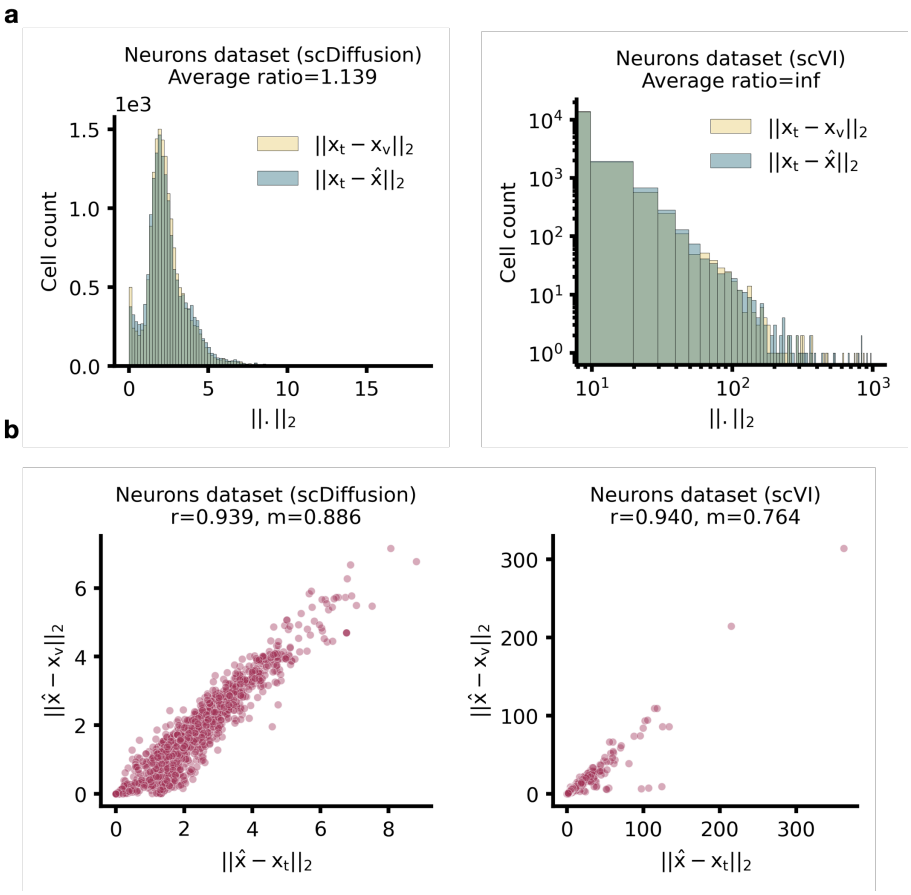

Figure A13. **Memorisation checks of scDiffusion and scVI on the neurons dataset**. Distribution of the Euclidean distance between the real cells in the training dataset x_t and the real cells in the validation dataset x_v (yellow) and the Euclidean distances between the training dataset and the generated cells $\hat{x}$ (blue) for scDiffusion (a, left) and scVI (a, right). Euclidean distances of the generated cells between the training dataset versus Euclidean distances of the generated cells to the validation dataset for scDiffusion (b, left) and scVI (b, right). r corresponds to the Pearson correlation coefficient and m corresponds to the slope of a linear fit.

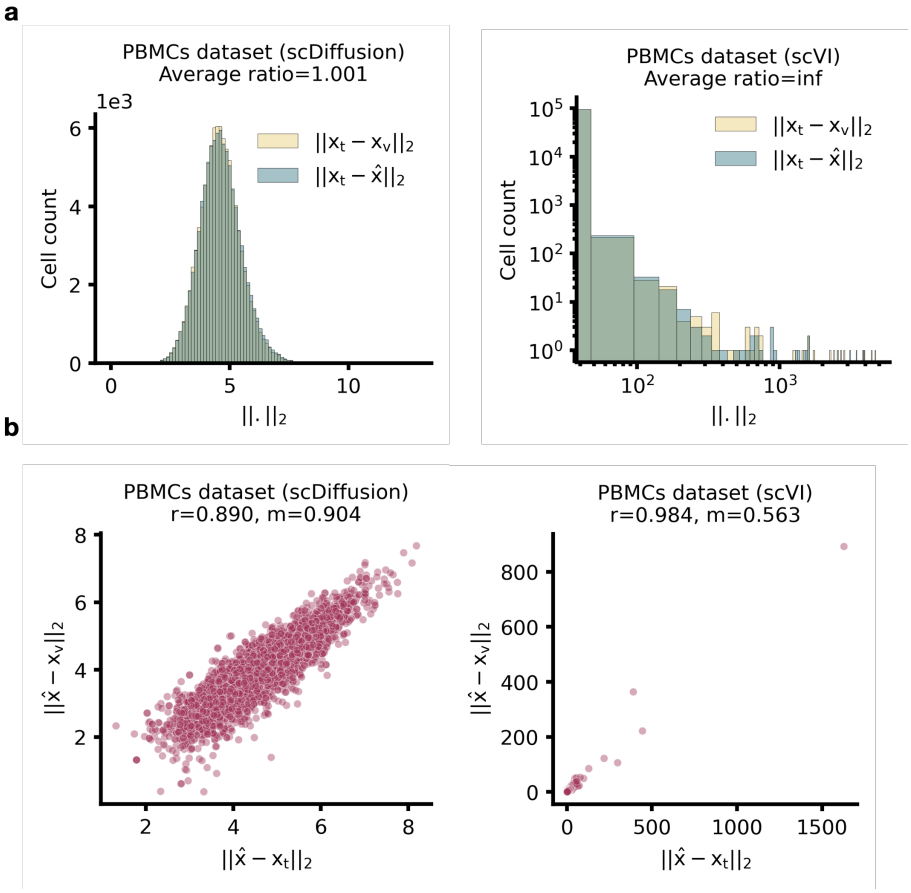

Figure A14. **Memorisation checks of scDiffusion and scVI on the PBMCs dataset**. Distribution of the Euclidean distance between the real cells in the training dataset x_t and the real cells in the validation dataset x_v (yellow) and the Euclidean distances between the training datset and the generated cells $\hat{x}$ (blue) for scDiffusion (a, left) and scVI (a, right). Euclidean distances of the generated cells between the training dataset versus Euclidean distances of the generated cells to the validation dataset for scDiffusion (b, left) and scVI (b, right). r corresponds to the Pearson correlation coefficient and m corresponds to the slope of a linear fit.

A3.4. Evaluating C2ST performance

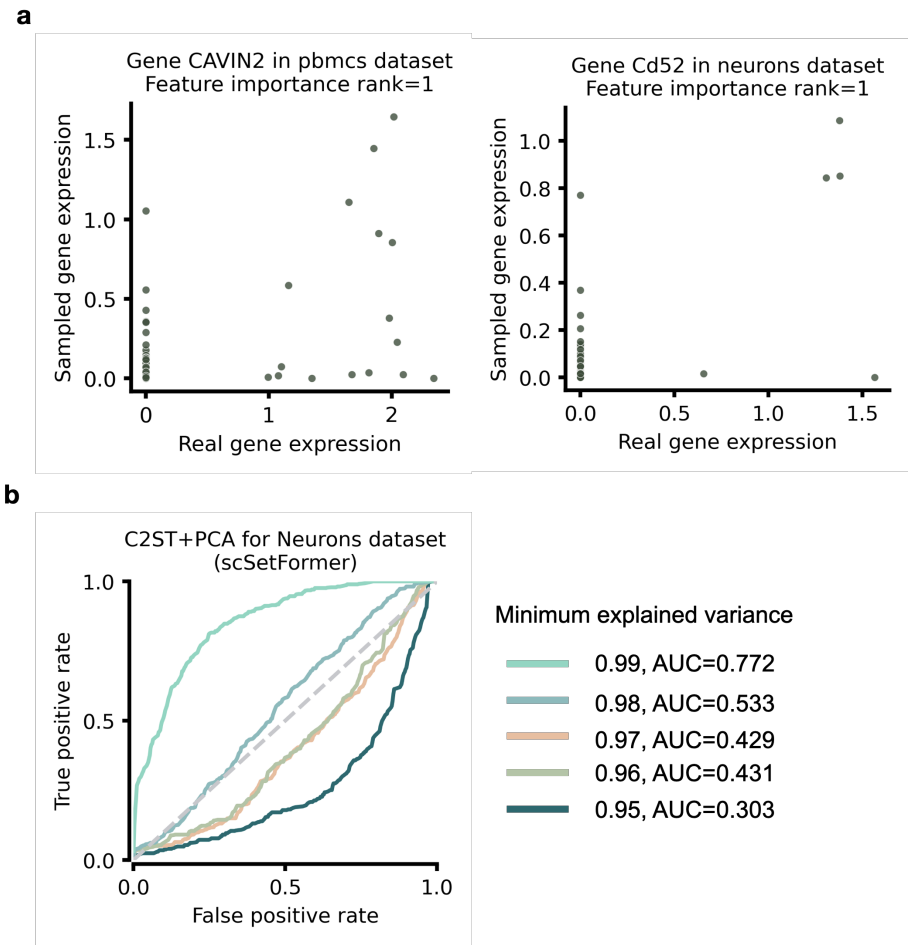

Figure A15. **Evaluation of C2ST.** Real vs. sampled expression values of the gene with the highest feature importance for the C2ST when applied to the PBMCs (a, left) and neurons (a, right) datasets. (b) C2ST applied to the transformed real and sampled neurons dataset by scSetFormer with 5 increasing minimum explained data variance to detect the top principal components.

A3.5. Gene set enrichment analysis

Table A1. **Gene set enrichment analysis for the PBMCs dataset.** Top 5 gene pathways that are significantly enriched in the real NK T cells by scSetFormer based on the top ten significantly upregulated differential genes. The first number is the number of upregulated genes in the real NK T cells and the second number is the number of genes in the respective pathway.

Gene pathway	Gene overlap
Cellular response to interferon-gamma	5/121
Inflammatory response	4/230
Response to interleukin-1	3/86
Response to interferon-gamma	3/80
Neutrophil migration	3/77

Table A2. **Gene set enrichment analysis for the PBMCs dataset.** Top 5 gene pathways that are significantly enriched in the generated NK T cells by scSetFormer based on the top ten significantly upregulated differential genes. The first number is the number of upregulated genes in the generated NK T cells and the second number is the number of genes in the respective pathway.

Gene pathway	Gene overlap
Cellular response to cytokine stimulus	5/482
Inflammatory response	4/230
Cellular response to interferon-gamma	4/121
Response to interleukin-1	3/86
Response to interferon-gamma	3/80

Table A3. **Gene set enrichment analysis for the PBMCs dataset.** Top 5 gene pathways that are significantly enriched in the real CD4 CTL cells by scSetFormer based on the top ten significantly upregulated differential genes. The first number is the number of upregulated genes in the real CD4 CTL cells and the second number is the number of genes in the respective pathway.

Gene pathway	Gene overlap
Positive thymic T cell selection	2/5
Negative regulation of lipid storage	2/20

Table A4. **Gene set enrichment analysis for the PBMCs dataset.** Top 5 gene pathways that are significantly enriched in the generated CD4 CTL cells by scSetFormer based on the top ten significantly upregulated differential genes. The first number is the number of upregulated genes in the generated CD4 CTL cells and the second number is the number of genes in the respective pathway.

Gene pathway	Gene overlap
Membrane organisation	3/242
Positive thymic T cell selection	2/5

Table A5. **Gene set enrichment analysis for the PBMCs dataset.** Top 5 gene pathways that are significantly enriched in the real CD8 TCM cells by scSetFormer based on the top ten significantly upregulated differential genes. The first number is the number of upregulated genes in the real B memory cells and the second number is the number of genes in the respective pathway.

Gene pathway	Gene overlap
Membrane organisation	3/242
Regulation of immune response	3/179
Positive thymic T cell selection	2/5

Table A6. **Gene set enrichment analysis for the PBMCs dataset.** Top 5 gene pathways that are significantly enriched in the generated CD8 TCM cells by scSetFormer based on the top ten significantly upregulated differential genes. The first number is the number of upregulated genes in the generated CD8 TCM cells and the second number is the number of genes in the respective pathway.

Gene pathway	Gene overlap
Positive thymic T cell selection	2/5

Table A7. **Benchmarking metrics on the PBMCs dataset in the multi-conditional mode.** MMD: maximum mean discrepancy, SWD: sliced-Wasserstein distance, C2ST: classifier two-sample test.

Model	MMD	SWD	C2ST	C2ST+PCA
Baseline	7.868-06	0.044	0.501	0.501
scSetFormer	2.041e-04	2.078	0.999	0.975

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Tübingen, den 09.04.2024 Unterschrift